

Proceedings of the 15th International Conference
on the Practice and Theory of Automated Timetabling

August 25–28, 2026
University of Nottingham
Nottingham, UK

ISBN: 978-0-9929984-7-9

Edited by
Daniel Karapetyan
John Drake
Ender Özcan

Preface

This volume contains the full papers, extended abstracts and demonstration abstracts presented at the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2026), held from 25 to 28 August 2026 in Nottingham, United Kingdom. The conference attracted 82 delegates, including 29 students, from 26 countries across six continents.

PATAT is a biennial conference dedicated to the theory and practice of computer-aided timetable generation. It serves as the main forum for the EURO Working Group on the Practice and Theory of Automated Timetabling (EWG-PATAT), an international community of researchers, practitioners and vendors.

This volume includes 22 full papers and 32 extended and demonstration abstracts. Each contribution was reviewed by selected members of the programme committee members and/or external reviewers in a single-blind process. The contributions span a broad range of timetabling, scheduling, and optimisation themes, including university course timetabling, school and educational planning, healthcare scheduling, production and manufacturing scheduling, personnel scheduling, routing and logistics, and sports scheduling.

The volume also includes the abstracts for the four invited talks. Hana Rudová (Masaryk University, Brno, Czech Republic) gave her talk on *Elegance of Simplicity in Designing Metaheuristics for Real-World Applications*. Kevin Tierney (University of Vienna, Vienna, Austria) discussed *Automated Heuristic Discovery: From Deep Reinforcement Learning to Large Language Models*. David Pisinger (Technical University of Denmark, Lyngby, Denmark) spoke on *Stochastic Adaptive Large Neighbourhood Search for Two-Stage Stochastic Routing and Scheduling Problems*. Jeroen Mulder (Air France-KLM, France) gave a talk on *Resilient Airline Planning: From Fleet Planning to Operations Control*.

We thank the authors, the invited speakers, the session chairs, the programme committee, the external reviewers, the local organisers and the steering committee for their contributions to the success of this conference. We also thank the sponsors of PATAT 2026 for their support: Eventmap, EWG-PATAT and the School of Computer Science, University of Nottingham.

Daniel Karapetyan
John Drake
Ender Özcan
PATAT 2026 Programme Chairs
Nottingham, United Kingdom, October 2026

Organisation

Programme Committee Chairs

Daniel Karapetyan	University of Nottingham (UK)
John Drake	University of Leicester (UK)
Ender Özcan	University of Nottingham (UK)

Local Organisers

Daniel Karapetyan	University of Nottingham (UK)
John Drake	University of Leicester (UK)
Daniel Torres Ruiz	University of Nottingham (UK)
Nilgun Sengoz	University of Nottingham (UK)

Steering Committee

Edmund K. Burke *(Honorary Chair)*	University of Leicester (UK)
Barry McCollum *(Treasurer)*	EventMAP (UK)
Patrick De Causmaecker	Katholieke Universiteit Leuven (BE)
Daniel Karapetyan	University of Nottingham (UK)
Jeffrey H. Kingston	University of Sydney (AU)
Nysret Musliu	Vienna University of Technology (AT)
Ender Özcan *(Chair)*	University of Nottingham (UK)
Hana Rudova	Masaryk University (CZ)
Andrea Schaerf	University of Udine (IT)
Thomas Jacob Riis Stidsen	Technical University of Denmark (DK)
Greet Vanden Berghe *(EWG-PATAT liaison)*	Katholieke Universiteit Leuven (BE)

Programme Committee

Ahmed Kheiri	Lancaster University (UK)
Aldy Gunawan	Singapore Management University (SG)
Alexandre Lemos	Outsystems (PT)
Andrea Schaerf	University of Udine (IT)
Andrew Parkes	University of Nottingham (UK)
Daniel Karapetyan	University of Nottingham (UK)

David Van Bulck	Ghent University (BE)
Dries Goossens	Ghent University (BE)
Efstratios Rappos	HEIG-VD (CH)
Elina Rönnberg	Linköping University (SE)
Ender Ozcan	University of Nottingham (UK)
Erwin Pesch	University of Siegen (DE)
Federico Della Croce	DIGEP Politecnico di Torino (IT)
Gerald Lach	MathPLan GmbH (DE)
Gerhard Post	University of Twente (NL)
Günther Raidl	Vienna University of Technology (AT)
Greet Vanden Berghe	KU Leuven (BE)
Hana Rudová	Masaryk University (CZ)
Hesham Alfares	King Fahd University of Petroleum & Minerals (SA)
Jeffrey H. Kingston	The University of Sydney (AU)
John H. Drake	University of Leicester (UK)
Jonathan Thompson	Cardiff University (UK)
Kadri Sylejmani	University of Prishtina
Luca Di Gaspero	University of Udine (IT)
Marco Chiarandini	University of Southern Denmark (DK)
Marco Lübbecke	RWTH Aachen University (DE)
Mats Carlsson	Research Institutes of Sweden (SE)
Michel Gendreau	Polytechnique Montréal (CA)
Nysret Musliu	TU Wien (AT)
Panayiotis Alefragis	University of Peloponnese (GR)
Paolo Toth	University of Bologna (IT)
Patrick De Causmaecker	KU Leuven (BE)
Pieter Smet	KU Leuven (BE)
Rhydian Lewis	Cardiff University (UK)
Sara Ceschia	University of Udine (IT)
Thomas Stidsen	Technical University of Denmark (DK)
Thomas Stützle	Université Libre de Bruxelles (BE)
Tomas Muller	Purdue University (US)
Túlio Angelo Machado Toffolo	KU Leuven (BE)

Sponsors



EventMap



EURO Working Group
on Automate Timetabling



School of Computer Science,
University of Nottingham

Table of Contents

Part I. Invited Talks 1
Stochastic ALNS for Two-Stage Stochastic Routing and Scheduling Problems
David Pisinger 2
Elegance of Simplicity: Designing Metaheuristics for Real-World Applications
Hana Rudová 3
Toward Resilient Airline Planning: From Fleet Planning to Operations Control
Jeroen Mulder 4
Automated Heuristic Discovery: from Deep Reinforcement Learning to Large Language Models
Kevin Tierney 5

Part II. Full Papers 6
Automated Verification of Pruning Rules for Runway Sequencing
Toby Clark, Jason Atkin and Geert De Maere 7
Iterated Local Search for the Trigger Arc Travelling Salesman Problem
Admir Kadriu and Kadri Sylejmani 19
An Iterative Model Reduction Method and a Constraint Programming Formulation for the Train Dispatching Problem
László Kálmán Trautsch and Bence Kovari 31
Extending the Minimal Perturbation Problem with Disruptiveness Weighting for Dynamic University Timetabling
Helen Kirby-Hawkins 45
Personnel Rostering in a Public Library: A Case Study
Bruno Chinaglia, Douglas Nascimento, Greet Vanden Berghe and Franklina Toledo 55
Integrative Hybrid Local Search with a Restart Strategy for University Course Timetabling with Student Sectioning
Mingxuan Li, Thomas Weise, Sina Abdipoor, Jourdan D'Orville, Say Leng Goh, Razali Yaakob and Salwani Abdullah 67
A Genetic Algorithm with Surrogate-Assisted Dynamic Programming for Maintenance Constrained Parallel Machine Scheduling
Fatima Iken, Abdennour Azerine, Mahmoud Golabi and Lhassane Idoumghar 81
Synthetic Instance Generation for the Radiotherapy Scheduling Problem
Chiara Camilla Rambaldi Migliore, Nysret Musliu, Giovanni Iacca and Marco Roveri 93

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Modular Constraint Injection from Natural Language for CP-Based Scheduling
Florian Strohm, Patrick Wagner, Jannik Schwab and Marco Huber ... 105
Reasoning about University Study Regulations Module-based Student Constraints and Preferences in Answer Set Programming
Henry Otunuya and Sebastian Schellhorn ... 121
Configuration Support for Lexicographic Multi-Objective Scheduling: A Framework Applied to Electric Vehicle Charging
Laurenz Tomandl, Günther Raidl, Tobias Rodemann and Steffen Limmer ... 137
MounouZinc: A MiniZinc-Based University Timetabling Tool Integrating AI Optimization and Web Automation
Tony Rouhayem, Youssef Bakouny, Laurent Drapeau and Melhem El Helou ... 149
Fast roster generation for single-nurse rostering problems
Robin Tourlamain, Pieter Smet and Greet Vanden Berghe ... 161
Assignment and Routing in the Home Health Care Problem
Shihhung Su and Chiu-Yuan Chen ... 173
Instance Space Analysis for Modular High School Timetabling
Andreas Krystallidis, Rubén Ruiz-Torrubiano and Nysret Musliu ... 187
Establishing a National Academic Timetabling Coordination Framework: The Case of the Forum for Tanzania Academic Timetable Coordinators (FTATC) in Tanzania
Neema Kumburu, Alfred Nchimbi and Victoria Kessy ... 201
Weighted Spearman Phenotypic Characterization for Surrogate-Assisted Genetic Programming in Dynamic Vehicle Scheduling
Leshan Tan, Chenwei Jin, Xinan Chen, Rong Qu and Ruibin Bai ... 219
Constraint Learning and Schedule Proposal via Transformers for the Nurse Scheduling Problem
Walid Abdelaidoum, Mohamed A. Madani, Larbi Boubchir and Boubaker Daachi ... 231
Data-Driven Merge Point Selection in Dynamic Aircraft Arrival Routes Optimization
Roghayeh Hajizadeh, Maksym Moroz, Tatiana Polishchuk, Elina Rönnberg and Christiane Schmidt ... 235
Mathematical Techniques for Evaluating University Timetabling Policies
Matthew Davison, Ahmed Kheiri and Konstantinos G. Zografos ... 249
Reentrant Permutation Flow Shop Problems with the Total Completion Time Objective
Itamar Segal, Tal Grinshpoun, Hagai Ilani and Elad Shufan ... 263
Improved bounds and solutions for the invigilator assignment problem
Reinout Annaert ... 275

Part II. Extended Abstracts and Demos ... 287

Evaluating Predict-then-Optimize methods for surgical case planning
Pieter Smet and David Kirpekar-Sauer 289
Decomposition Techniques for the Aircraft Maintenance Routing Problem
Ida Gjergji, Lucas Kletzander, Hendrik Bierlee, Nysret Musliu and Peter J. Stuckey 293
KHE26: Progress towards an Efficient, Practical Nurse Rostering Solver (Abstract)
Jeffrey Kingston 297
Educational Timetabling with Unconstrained Demand Forecasting
Olivia Phillips, Dima Nazzal and Joel Sokol 301
Novel neighborhood structures for incomplete round robin tournaments
Karel Devriesere, David Van Bulck and Dries Goossens 307
Case Study: An Integrated Airport Security Planning Approach Using Simulated Annealing and Minimum Cost Flow
Hicham Kamhi and Lotte Berghman 311
A Hybrid Adaptive Metaheuristic-based Q-Learning Approach for the Home Healthcare Routing and Scheduling Problem
Yun-Chia Liang, Angela Hsiang-Ling Chen, Sebastian Gunawan, Vanny Minanda 315
Branch-Price-and-Cut and Symmetry Reductions for Sports Tournament Scheduling
Jasper van Doornmalen and David Van Bulck 321
Multi-Contract Integrated Shift Design and Rostering
Lucas Kletzander 325
Large Language Model-Based Explainable Hyper-heuristics for Examination Timetabling
Nilgun Sengoz, Ender Ozcan and Jeremie Clos 329
Large Neighborhood Search for Generalized Parallel Machine Scheduling
Lukas Frühwirth, Nysret Musliu and Felix Winter 337
Untis Course Optimization for Finnish High Schools
Lukas Wolf 343
Instructor Scheduling in UniTime
Tomas Muller 345
Scheduling with No-Wait Decisions in Heavily Loaded TSN Networks
Jesper Vines, Hannes Uppman, Mikaela Lindberg and Elina Rönnberg 351
A Demonstration of an Integrated Project Management and Demo Scheduling Platform
Neel Gaglani and Ender Özcan 355
A Bandit-Based Reinforcement Learning Simulated Annealing for the Omnichannel Last-Mile Distribution
Vincent F. Yu, Renan S. Maglasang, Aldy Gunawan, Ching-Hsuan Lin, Kuan-Fu Chen and Shih-Wei Lin 365

New Insights into Local Search for Curriculum-Based Course Timetabling
Sara Ceschia, Francesca Da Ros, Luca Di Gaspero and Andrea Schaerf 371

Adaptive Heterogeneous Sand Cat Swarm Algorithm for Single-Objective Flexible Job Shop Scheduling
Wirawati Dewi Ahmad, Azuraliza Abu Bakar and Mohd Nor Akmal Khalid 375

Explaining Infeasibility in the Test Laboratory Scheduling Problem
Aida Aliu, Florian Mischek and Nysret Musliu 381

Fairness-aware opponent assignment in sports timetabling with fixed home dates
Ali Hassanzadeh 385

Fair allocation of over- and understaffed shifts in nurse rostering
Lisa Garcia Tercero, Pieter Smet, Dries Goossens and Greet Vanden Berghe 391

The Incomplete Traveling Tournament Problem
David Van Bulck, Karel Devriesere and Dries Goossens 395

Refining self-scheduled rosters with limited changes and blocks
Qing Chuan Ye (registered as Charlie Ye) 399

Online Outperforms Offline: Reinforcement Learning Hyper-Heuristics for Vehicle Routing
Daniel Torres Ruiz, Daniel Karapetyan, Ender Ozcan and Rong Qu . . 403

From the Field between Theory and Practice in Scheduling
Gerhard Post 407

Including Human Centered Objectives into Train Crew Rostering Algorithms
Veerle van den Hurk, Erwin Abbink, Marjan van den Akker and Egon L. van den Broek 411

Simultaneous Optimization of Two-Shift and Three-Shift Nurse Rosters with Infection-Risk-Based Transition
Mio Hourai, Wei Wu and Mutsunori Yagiura 417

A Three-Phase Framework for Course Planning and Timetabling in Upper Secondary Schools with Self-Directed Learning
Michael Hölscher 421

Smart TV Scheduling in Public Venues: A New Combinatorial Problem and Optimization Approaches
Shefket Bylygbashi, Fisnik Hazrolli, Endrita Vllasaliu, Eljon Shala, Yll Berisha, Kaltrina Krasniqi, Kadri Sylejmani, Labeat Arbneshi and Uran Lajqi 425

Mathematical Optimization of the Rail Infrastructure Planning Timetable
Alexander Heide Madsen, Victor Alexander Brandsen and Thomas Stidsen 435

The Integrated Healthcare Timetabling Competition 2024: Insights, Benchmarking and Algorithmic Trends
Sara Ceschia, Roberto Maria Rosati, Andrea Schaerf, Pieter Smet, Greet Vanden Berghe and Eugenia Zanazzo 441
An Iterated Local Search Approach for One-Sided Crossing Minimization
Labeat Arbneshi and Kadri Sylejmani 445

Keyword Index 449

Author Index 452

Part I
Invited Talks

Stochastic ALNS for Two-Stage Stochastic Routing and Scheduling Problems

David Pisinger[1]

Technical University of Denmark, Lyngby, Denmark
dapi@dtu.dk

Abstract. We present a general local search framework for scenario-based two-stage combinatorial stochastic programming problems. The framework is based on Adaptive Large Neighborhood Search (ALNS), where an upper-level local search method operates on the first-stage decision variables, while a number of lower-level local search methods optimize the individual scenarios. As a demonstration of the Stochastic ALNS algorithm, the framework is tested on the two-stage stochastic Prize-collecting Vehicle Routing Problem, and stochastic Team Orienteering Problem. In all problems, the customers, demands and travel times can be stochastic as long as they can be represented by a number of scenarios. Computational experiments show that the algorithm scales well with the number of scenarios, making it possible to solve instances with up to 1000 customers in short time.

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Elegance of Simplicity: Designing Metaheuristics for Real-World Applications

Hana Rudová[1]

Masaryk University, Brno, Czech Republic
hanka@fi.muni.cz

Abstract. Metaheuristics serve as the primary building blocks for modern industrial solvers and complex optimization engines. While their flexibility allows for rapid integration into both academic frameworks and practical software, the pursuit of performance often leads to their over-engineered designs. Beyond that, we emphasize the idea of simplicity and simplification in algorithm design as the keys to generality and robustness. Drawing on a wide range of successful applications, we show how distilling algorithms to their core principles leads to better outcomes. We present case studies across several optimization challenges: implementing slack induction by string removals in vehicle routing, utilizing iterative forward search in course timetabling, and applying local search in grid scheduling and warehouse planning. Furthermore, we discuss how cross-domain hyper-heuristics benefit from this philosophy. With a focus on robust performance across diverse and complex domains, designing simple(r) algorithms can advance the current state of the art.

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Toward Resilient Airline Planning: From Fleet Planning to Operations Control

Jeroen Mulder[1]

Air France-KLM, France

Abstract. Airlines make thousands of interconnected decisions across different planning horizons, ranging from long-term fleet and network planning to real-time operational control. While each planning process has traditionally been optimized in isolation, increasing operational complexity and uncertainty call for a more integrated perspective on resilience. Drawing on insights from an ongoing research collaboration, initiated through the Kickstart.AI program, between Air France-KLM, Utrecht University, and the University of Twente, this presentation places a series of research projects within the broader context of the airline planning process. Through practical case studies, it highlights three recurring themes: accounting for interdependencies across planning processes, explicitly incorporating uncertainty into planning models, and developing customized optimization methods tailored to the unique characteristics of airline planning.

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Automated Heuristic Discovery: from Deep Reinforcement Learning to Large Language Models

Kevin Tierney[1]

University of Vienna, Vienna, Austria
kevin.tierney@univie.ac.at

Abstract. Designing high-performance heuristics is a complex task that requires both intuition and deep domain knowledge. Recent advances in deep reinforcement learning (DRL) and large language models (LLMs) have led to new ways of creating heuristics that rival the performance of state-of-the-art human-designed approaches. In this talk, I introduce these new mechanisms for heuristic design and describe their advantages and limitations, with a focus on routing and scheduling problems. The talk will include two methods: Neural Deconstruction Search, an approach harnessing DRL to learn policies for constructing routing and scheduling solutions, and VRPAgent, an evolutionary approach to generate novel routing heuristics.

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Part II
Full Papers

Automated Verification of Pruning Rules for Runway Sequencing

Toby Benjamin Clark, Jason Atkin, and Geert De Maere

Computational Optimisation and Learning Lab,
School of Computer Science, University of Nottingham, UK
{toby.clark, jason.atkin, geert.de_maere}@nottingham.ac.uk

Abstract. Exact approaches to the Runway Sequencing Problem (RSP) rely on pruning rules to make optimisation tractable. However, these rules are typically proven manually, making it difficult to evaluate new candidates or reuse rules across problem variants. We propose the use of Satisfiability Modulo Theories to automatically verify pruning rules for the RSP. Our approach verifies several published pruning rules within a symbolic sequence abstraction, where correctness is established by refuting all counterexamples. This enables a more systematic workflow for pruning rule development, and provides a basis for future rule synthesis.

Keywords: Runway Sequencing · Pruning Rules · Verification · Exact Optimisation · Satisfiability Modulo Theories (SMT) · Formal Methods

1 Introduction

Runway Sequencing is an important online timetabling problem at congested airports, where the efficient use of runways has a direct impact on delay propagation, emissions, and operational costs. Despite the NP-hardness of the general formulation, practical runway sequencing instances possess structural characteristics in their separation constraints and objective functions that can be systematically exploited, enabling the application of exact solution methods [8].

One approach to solving such instances defines a set of pruning rules [5], which eliminate partial sequences that are provably dominated by other partial sequences. These leverage structural characteristics of the problem constraints to reduce average time complexity, but rely on manual proofs to certify they preserve optimal solutions. However, manual reasoning is tightly coupled to a specific formulation, making rules difficult to reuse across variants with differing constraints or assumptions. Even small model changes can invalidate a pruning rule, necessitating a human review to re-prove rule correctness under the changes.

We propose the use of Satisfiability Modulo Theories (SMT) for verifying the correctness of such rules. SMT extends Boolean satisfiability with constructs such as linear arithmetic to enable automated verification of pruning rules, including those introduced in [5]. Specifically, we use the Z3 Theorem Prover [7] to provide the first machine-executable formalisation showing that such rules preserve correctness, and demonstrate how runway sequencing constraints can be translated into a logically verifiable representation, and checked programmatically.

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Pruning Rules in Optimisation. Many optimisation algorithms (e.g. dynamic programming, branch-and-bound, tree search) incrementally construct partial solutions and extend them until a complete solution is found. Pruning rules define dominance relations over partial solutions, enabling the early elimination of inferior search branches without compromising optimality.

Whilst pruning rules are not optimisers in themselves, a single correctness proof is orthogonal to the choice of search strategy, enabling their safe reuse across many search procedures that operate over partial sequences.

Pruning Runway Sequencing. De Maere et al. [5] introduced a set of pruning rules for runway sequencing, reducing average time complexity from $O(n!)$ to $O(N^2(n+1)^N)$ without loss of optimality, where n is the number of aircraft, and N is the number of distinct aircraft types. Such rules have been applied to instances based on sequencing systems at London Heathrow Airport [5], highlighting their practical relevance and applicability to realistic problem instances.

These rules are derived from structural properties of wake-separation constraints and time window feasibility, and induce a dominance relation over partial sequences. More broadly, pruning rules have received considerable attention in machine scheduling literature, often used in exact methodologies [1, 2].

Logical Structure & Correctness. Pruning rules can be viewed as logical implications: their preconditions specify when the rule may be applied to a partial solution, and their dominance claim states that the pruned branch cannot yield a better feasible completion with respect to some objective component.

Rules are correct if this implication holds for all feasible instances. Equivalently, whenever the preconditions are satisfied, every feasible completion of the pruned branch is dominated by some feasible completion of the retained branch.

Contributions. We propose the use of Satisfiability Modulo Theories to automatically verify pruning rules for the Runway Sequencing Problem (RSP), including several rules introduced in [5]. In addition to the theoretical contributions outlined below, we implement the approach using the Z3 Theorem Prover and provide executable notebooks that reproduce all verification results.

- We formalise the implicit and explicit semantics of the RSP in SMT.
- We develop a refutation-based SMT approach for verifying pruning rules.
- We automatically verify several of the pruning rules introduced in [5].
- We propose a counterexample-preserving abstraction for runway sequencing.

Our contribution is therefore not a new pruning rule, but a reusable SMT formulation for automatically verifying complete and conditional order pruning rules for the RSP. The approach is intended as an offline verification tool rather than an operational runtime component or evaluation of pruning rule effectiveness.

Paper Overview. This paper proceeds as follows: Section 2 introduces SMT; Section 3 introduces the operational constraints of the RSP; Section 4 introduces the verification encodings and Section 5 evaluates the approach.

2 SMT for Pruning Rule Verification

Given that pruning rules are correctness properties rather than performance or approximation claims, validating them amounts to proving a safety property of the search, namely an optimality-preserving invariant over the search space.

However, as correctness is universally quantified, empirical testing can at best reveal counterexamples; it cannot establish validity across all possible configurations. In the RSP, the state space is infinite, as many decision variables range over the real numbers, making exhaustive enumeration impossible.

Therefore, the correctness of pruning rules must be derived logically (or inductively) from the problem formulation. SMT solvers enable this by combining SAT-based reasoning with specialised theory solvers (e.g. linear arithmetic, typically handled via simplex-style methods), allowing them to reason over highly expressive constraint systems without the need for explicit enumeration.

Manual Proof. Correctness arguments in [5] are established manually, typically appealing inductively to the monotonic structure (with respect to takeoff times) of the delay cost, makespan, and regulatory penalties defined in Section 3.

Limitations. However, manual reasoning is brittle under model evolution, as correctness arguments are formulation-specific and lack the structural invariance required to adapt across variants of the Runway Sequencing Problem.

Moreover, the arguments are non-executable, differ rule-by-rule, and offer no generic mechanism for verification, reuse, or quickly testing new rule candidates. Consequently, embedding these rules into new problem variants requires time-consuming rederivation, which can be considered a barrier to their adoption.

SMT. Satisfiability Modulo Theories enables automated, symbolic reasoning over constraints in rich background theories[1], and has become a standard tool for machine-checked verification of correctness properties in programs [7]. Given a set of constraints expressed in first-order logic over background theories, an SMT solver decides their satisfiability and, when satisfiable, can construct a corresponding instance, i.e. a finite assignment of values to variables.

The Z3 Theorem Prover [7] is an SMT solver, with applications in both program verification [3] and operational research [6]. In this work, SMT is not used to compute or optimise runway sequences, only to verify pruning rules.

Pragmatic Benefits. Automated verification of pruning rules supports trustworthy exact search, enables quick rechecking when the problem formulation changes, and facilitates iterative development and refinement of new rules.

Furthermore, SMT encodings are lightweight to develop. Modern SMT solvers, such as Z3, provide bindings for several programming languages, allowing verification constraints to be constructed and executed programmatically.

[1] Solvers for SMT support reasoning over highly expressive extensions of first-order logic, such as linear real/integer arithmetic, nonlinear arithmetic (sometimes undecidable), bit-vectors, arrays, algebraic datatypes, and uninterpreted functions [7].

3 Runway Sequencing Problem

Given a set of aircraft $\mathcal{A}$, De Maere et al. [5] define the Runway Sequencing Problem as the task of determining a total order of takeoffs that satisfies asymmetric minimum separations between aircraft (due to wake vortices and airspace restrictions), and strict feasibility windows, while minimising operational delay, makespan, and regulatory penalties. Formally, a runway sequencing instance $\mathcal{P}$ is defined by the tuple of vectors/matrices shown by Equation (1).

$$\mathcal{P} = \big(\mathcal{A},\ \delta,\ \langle ec, lc\rangle,\ \langle et, lt\rangle,\ b,\ c\big) \tag{1}$$

Here, for each aircraft $i \in \mathcal{A}$, b_i denotes the earliest time at which i can enter the runway queue, c_i is a fixed queuing delay, $\langle et_i, lt_i\rangle$ is a hard feasibility window on takeoff time, and $\langle ec_i, lc_i\rangle$ is a CTOT[2] window. For any ordered pair of aircraft $(i, j) \in \mathcal{A} \times \mathcal{A}$, δ_{ij} denotes the minimum temporal wake separation required if aircraft i precedes j, and is generally asymmetric. All timing variables are assumed to be continuous and measured in seconds.

Takeoff and Release. Given a sequence S over $\mathcal{A}$, let $S_i \subset \mathcal{A}$ denote the set of aircraft preceding i in S. Assuming aircraft depart as early as operationally feasible (a reasonable assumption at busy airports with takeoff queues), the takeoff time (2) of aircraft i is given as the latest of its release time, which is the earliest possible time an aircraft can be sequenced (3), and the minimum δ-separation from all aircraft preceding i in sequence S.

$$t_i = \max\big(r_i,\ \max_{x \in S_i}(t_x + \delta_{xi})\big) \tag{2}$$

$$r_i = \max(b_i + c_i,\ et_i,\ ec_i) \tag{3}$$

Objective Function. Orderings are ranked in [5] using a multi-objective function (4). The first objective measures makespan, defined as the maximum takeoff time in sequence S. The second objective measures delay, computed as the sum over aircraft of weighted deviations between base times and scheduled takeoff times (where α penalises greater delays more, promoting equity), together with a linear penalty for aircraft that miss their CTOT slots (5).

$$F(S) = \left(\max_{\ell \in S} t_\ell,\ \sum_{i \in S}\big(W_1\,(t_i - b_i)^\alpha + W_2\,C(t_i, lc_i)\big)\right) \tag{4}$$

CTOT Penalty. Violations of CTOT windows are penalised using a weighted piecewise linear function (5) with operator-defined parameters (ω_1 to ω_4).

$$C(t_i, lc_i) = \begin{cases} 0, & t_i \le lc_i, \\ \omega_1\,(t_i - lc_i) + \omega_2, & lc_i < t_i \le lc_i + 300, \\ \omega_3\,(t_i - lc_i) + \omega_4, & t_i > lc_i + 300. \end{cases} \tag{5}$$

[2] CTOT (Calculated Takeoff Time) is a time slot allocated by air traffic flow management to balance network demand, with penalties applied for late departure.

4 SMT Formalisation and Verification

We formalise the core feasibility constraints of the RSP as SMT-solvable encodings. These are expressed in quantifier-free linear real arithmetic[3] with if-then-else expressions. Polynomial objective terms introduced later induce a local theory extension to quantifier-free nonlinear real arithmetic (`QF_NRA`).

Timing Variables. All timing parameters are continuous and are therefore encoded over $\mathbb{R}$. Here, the notation $x : \tau$ denotes that variable x has type τ, since in SMT every variable must be declared with a fixed type. For each $i \in \mathcal{A}$, timing parameters are non-negative real-valued variables (6)–(7), with additional constraints enforcing total orders over time windows and CTOT slots (8).

$$r_i, b_i, c_i, ec_i, lc_i, et_i, lt_i \ : \mathbb{R},\ \forall i \in \mathcal{A} \qquad \delta_{xy} : \mathbb{R},\ \forall \langle x, y \rangle \in \mathcal{A} \times \mathcal{A} \tag{6}$$

$$r_i, b_i, c_i, ec_i, lc_i, et_i, lt_i \ \geq 0,\ \forall i \in \mathcal{A} \qquad \delta_{xy} \geq 0,\ \forall \langle x, y \rangle \in \mathcal{A} \times \mathcal{A} \tag{7}$$

$$et_i < lt_i \ \wedge \ ec_i < lc_i,\ \forall i \in \mathcal{A} \tag{8}$$

Release Times. The release time of aircraft i, defined in (3), is the maximum of its queue-adjusted base time, the start of its time window, and the start of its CTOT window. We encode this in SMT as (9). Whilst the max operator is non-native to SMT, it can be defined using nested if-then-else expressions.

$$r_i = \max(b_i + c_i,\ et_i,\ ec_i), \quad \forall i \in \mathcal{A} \tag{9}$$

Sequence Evaluation. Given a sequence S and aircraft $i \in S$, where S_i denotes the set of aircraft preceding i in S, the takeoff time is defined recursively by (10), which directly implements the operational takeoff semantics of (2).

$$t_i = \max\Big(r_i,\ \max_{x \in S_i}\big(t_x + \delta_{xi}\big)\Big) \tag{10}$$

Separation Identicality. Several rules in [5] apply to pairs of separation-identical aircraft. We introduce the binary relation $i =_\delta j$ to denote that aircraft i and j have identical separation requirements with respect to all aircraft in $\mathcal{A}$ (11).

$$i =_\delta j \ \underset{\text{DEF}}{\iff} (\delta_{ix} = \delta_{jx} \wedge \delta_{xi} = \delta_{xj}, \quad \forall x \in \mathcal{A}) \tag{11}$$

Weights & Constants. All operator weights and constants are non-negative real-valued parameters (12). Since α is an exponent, we constrain $\alpha \geq 1$. Several pruning rules rely on the monotonicity of the CTOT function (with respect to takeoff time). To ensure weightings preserve this property, we require (13).

$$W_1, W_2, \omega_1, \omega_2, \omega_3, \omega_4, \alpha : \mathbb{R} \qquad W_1, W_2, \omega_1, \omega_2, \omega_3, \omega_4 \ \geq 0 \qquad \alpha \geq 1 \tag{12}$$

$$(\omega_1 \leq \omega_3) \ \wedge \ (\omega_2 \leq \omega_4) \tag{13}$$

We fix α to a constant, since arbitrary exponentiation is generally undecidable in SMT. Similarly, $\omega_1 \ldots \omega_4$ and W_1, W_2 are treated as constants.

[3] In SMT, problems are classified by logical theories. `QF_LRA` allows linear constraints over $\mathbb{R}$ without quantifiers, while `QF_NRA` additionally permits nonlinear terms [7].

4.1 Proof by Refutation with SMT

Correctness arguments in [5] rely on showing that a dominance relation between two aircraft propagates inductively to all subsequent aircraft in the sequence, with respect to makespan, delay, time window feasibility, and CTOT penalty.

In contrast, our SMT-based approach does not reason inductively over sequence structure. Instead, each pruning rule is encoded as a global logical property over symbolic timing and separation parameters. Correctness is then established by ruling out the existence of any feasible counterexample.

Satisfiability & Proof. SMT solvers, such as Z3, check the satisfiability of logical formulae expressed over the supported background theories (see Sec. 2). If the formula is satisfiable, the solver returns an instance, i.e. a valid assignment of variables to values. Otherwise, if the formula is unsatisfiable, then the solver has automatically proven that no satisfying instantiation exists.

In SMT-based verification, the correctness of a hypothesis (e.g. that a pruning rule is correct) is established by proving that no counterexample exists. Assumptions are encoded together with the negation of the hypothesis; if this formula is unsatisfiable, then the claim is valid, as there is no instance in which all assumptions hold true and the hypothesis holds false.

In the context of pruning the RSP, we verify a pruning rule by showing that no feasible instance satisfies the rule preconditions whilst falsifying the dominance claim, i.e., satisfying the logical negation of the dominance claim.

Correctness Pattern. Let β be a pruning rule with precondition P and dominance claim Q, and let Φ denote the foundational constraints, defined in (6)–(13). Rule β is correct if, for all models satisfying Φ, P implies Q, (14).

$$(\Phi \wedge P) \;\Rightarrow\; Q. \tag{14}$$

Since SMT solvers decide satisfiability, rather than validity, directly asserting $\Phi \wedge P \Rightarrow Q$ and checking satisfiability would establish merely the existence of an instance in which β is correct, rather than its universally quantified correctness under all feasible configurations. Instead, we use the standard equivalences (15) and verify correctness by checking that $\Phi \wedge P \wedge \neg Q$ is unsatisfiable.

$$(\Phi \wedge P) \Rightarrow Q \;\equiv\; \neg(\Phi \wedge P) \vee Q \;\equiv\; \neg\big((\Phi \wedge P) \wedge \neg Q\big) \tag{15}$$

If $\Phi \wedge P \wedge \neg Q$ is unsatisfiable, then no configuration satisfies Φ and P while violating Q, therefore the pruning rule β is correct with respect to Φ. If satisfiable, a concrete counterexample is found, and can be used for debugging.

Non-Vacuity. Rules may appear correct simply because their preconditions are unsatisfiable. Consider $r_i \neq r_i$ as a rule precondition; it is a syntactically correct expression in a decidable theory, yet it is inherently unsatisfiable; thus any paired dominance claim would appear correct under naïve refutation alone.

We also require rules to be non-vacuous; verifying that $\Phi \wedge P$ is satisfiable guarantees the rule is able to prune in at least one feasible configuration.

4.2 Formulating Complete Order Pruning Rules

We demonstrate how various complete order pruning rules from [5] can be formulated and verified using SMT. The first rule is presented in detail, with the remaining rules following an analogous swapped-sequence formulation.

Complete order pruning rules state that aircraft i and j form a complete order if the objective value and feasibility of any sequence S including i and j cannot, under any circumstances, be improved by reversing i and j. Consequently, any sequence in which j precedes i can be safely pruned from the search.

Makespan Objective. In [5], it is shown that objectives to minimise makespan can induce a complete order upon two aircraft i and j if $r_i \leq r_j$ and $i =_\delta j$. We verify that, under these conditions, for any pair of sequences S, S' that differ only by swapping i and j, the makespan component in S is no worse than in S'.

Swapped Sequences. While concrete instances of S and S' may be arbitrary in length, we model counterexamples in aircraft preceding, between, and following i and j as abstract representative aircraft, later justified in Section 4.4.

Let S and S' be sequences that differ only by swapping the positions of i and j, and let ψ_1, ψ_2, ψ_3 denote abstract aircraft representing counterexamples before, between, and after i and j, with $\mathcal{A}$ defining the set of all aircraft (16).

$$S = \psi_1,\ i,\ \psi_2,\ j,\ \psi_3 \qquad S' = \psi_1,\ j,\ \psi_2,\ i,\ \psi_3 \qquad \mathcal{A} = \{\psi_1, \psi_2, \psi_3, i, j\} \tag{16}$$

Rule Formulation. An objective to minimise makespan is claimed to induce a complete order under the preconditions $r_i \leq r_j$ and $i =_\delta j$. Therefore, the formula $r_i \leq r_j \wedge i =_\delta j$ is encoded in SMT as the rule's precondition.

Makespan is defined in objective function (4) as the maximum takeoff time over a given sequence. Since S denotes the retained ordering, and S' denotes the pruned ordering, the rule claims that the makespan of S is always less than or equal to that of S', explicitly encoded in SMT as $\max_{\ell \in S}(t_\ell) \leq \max_{k \in S'}(t'_k)$.

Verifying Correctness. Following the pattern from Section 4.1, we encode in SMT the foundational constraints (Φ), in conjunction with the rule preconditions and the negation of the makespan dominance claim, shown explicitly in (17).

The SMT solver attempts to find an instance satisfying $r_i \leq r_j \wedge i =_\delta j$ whilst violating the claimed makespan dominance. If no such instance exists, then it follows that no counterexample exists, and the rule is correct.

$$\Phi \ \wedge \ \underbrace{(r_i \leq r_j \ \wedge \ i =_\delta j)}_{\text{Precondition}} \ \wedge \ \underbrace{\neg\big(\max_{\ell \in S}(t_\ell) \leq \max_{k \in S'}(t'_k)\big)}_{\text{Dominance Claim}} \tag{17}$$

When encoded in Z3, formula (17) is proven to be unsatisfiable, indicating that no counterexample exists; thus, the pruning rule is verified as correct[4].

Non-Vacuity. In addition to verifying correctness, we also check that the rule is non-vacuous; here we ensure that $\Phi \wedge (r_i \leq r_j \ \wedge \ i =_\delta j)$ is satisfiable.

[4] Full rule correctness and non-vacuity results are summarised later in Section 5.

Delay, CTOT & Time Windows. Let t_ℓ and t'_ℓ denote the takeoff times of aircraft ℓ in sequences S and S', respectively. For each aircraft $\ell \in S$, we define the delay cost D_ℓ, CTOT penalty C_ℓ, and time window violation V_ℓ in (18).

$$D_\ell = W_1(t_\ell - b_\ell)^\alpha \quad C_\ell = W_2 C(t_\ell,\ lc_\ell) \quad V_\ell = t_\ell > lt_\ell. \tag{18}$$

Here, $D_\ell, C_\ell : \mathbb{R}$ whilst V_ℓ is Boolean. Corresponding quantities for S' are denoted D'_ℓ, C'_ℓ, and V'_ℓ, defined analogously using the takeoff times t'_ℓ in S'.

Delay Objective. Objectives to minimise delay cost similarly induce a complete order between two aircraft i and j under the conditions $r_i \leq r_j$, $i =_\delta j$, and $b_i \leq b_j$. When these preconditions hold, sequencing i before j cannot yield a higher total delay cost than the swapped ordering in which j precedes i.

Rule Formulation. As with the encodings for makespan, we consider sequences S and S' defined in (16), and compare total delay over both sequences. Dominance between sequences S and S' with respect to total delay is defined in (19), where D_ℓ and D'_k denote the per-aircraft delay cost in S and S' respectively.

$$\sum_{\ell \in \mathcal{A}} (D_\ell) \leq \sum_{k \in \mathcal{A}} (D'_k) \tag{19}$$

To verify the rule, we encode the foundational constraints together with the rule preconditions and the negation of the total delay dominance claim (20).

$$\Phi \ \wedge \ (r_i \leq r_j \ \wedge \ i =_\delta j \ \wedge \ b_i \leq b_j) \ \wedge \ \neg\big(\sum_{\ell \in \mathcal{A}} (D_\ell) \leq \sum_{k \in \mathcal{A}} (D'_k)\big) \tag{20}$$

If this formula is unsatisfiable, then no instance satisfying $r_i \leq r_j, i =_\delta j$ and $b_i \leq b_j$ violates the dominance claim, and the rule is correct. We also verify that the rule is non-vacuous, ensuring $\Phi \wedge (r_i \leq r_j \wedge i =_\delta j \wedge b_i \leq b_j)$ is satisfiable.

Time Windows. Hard time windows can induce a complete order between two aircraft i and j whenever $r_i \leq r_j$, $i =_\delta j$, and $lt_i \leq lt_j$ hold. Unlike delay and makespan, this rule concerns feasibility rather than optimisation. Specifically, the claim is that if the swapped sequence S' is feasible (i.e. no aircraft violates its time window), then the original sequence S is also feasible.

Rule Formulation. Since V_ℓ and V'_ℓ denote ℓ missing its time window in S and S' respectively, the time window feasibility of sequence S can be modelled as the conjunction of $\neg V_\ell$ for each $\ell \in \mathcal{A}$, with S' defined analogously using V'_ℓ.

Given that the rule applies only when S' is feasible, we include the feasibility of S' into the preconditions. To verify the rule, we conjoin these preconditions with the foundational constraints and the negated dominance claim (21).

$$\Phi \ \wedge \ \Big(r_i \leq r_j \ \wedge \ i =_\delta j \ \wedge \ lt_i \leq lt_j \wedge \ \big(\bigwedge_{k \in \mathcal{A}} (\neg V'_k)\big)\Big) \wedge \ \neg\big(\bigwedge_{\ell \in \mathcal{A}} (\neg V_\ell)\big) \tag{21}$$

Here, Z3 searches for an instance where all preconditions hold, S' is feasible, and S is not. If unsatisfiable, no counterexample exists and the rule is correct.

4.3 Formulating Conditional Order Pruning Rules

Cost-minimising objectives can induce a conditional order between aircraft i and j when $r_i \leq r_j$, $lt_i \leq lt_j$, and $i =_\delta j$. Under these conditions, i should be sequenced before j whenever inequality $\mathcal{I}$ (22) holds, i.e., when the combined local cost of sequencing i before j is no greater than that of the swapped ordering.

$$\begin{aligned}\mathcal{I} = {} & W_1(t_i - b_i)^\alpha + W_2 C(t_i, lc_i) + W_1(t_j - b_j)^\alpha + W_2 C(t_j, lc_j) \\ \leq {} & W_1(t'_i - b_i)^\alpha + W_2 C(t'_i, lc_i) + W_1(t'_j - b_j)^\alpha + W_2 C(t'_j, lc_j)\end{aligned} \tag{22}$$

Unlike complete orders, conditional orders depend on takeoff times directly. We include inequality $\mathcal{I}$ in the rule preconditions and verify that sequencing i before j cannot worsen makespan, delay, or time window feasibility.

Rule Formulation. We present the formulation for makespan in (23). Encodings for delay and time window feasibility follow similarly from their complete order definitions, with the additional term $\mathcal{I}$ included in the preconditions.

$$\Phi \ \wedge \ \big(r_i \leq r_j \ \wedge \ i =_\delta j \ \wedge \ lt_i \leq lt_j \ \wedge \ \mathcal{I}\big) \ \wedge \ \neg\big(\max_{\ell \in S}(t_\ell) \leq \max_{k \in S'}(t'_k)\big) \tag{23}$$

Combined CTOT. Objectives penalising CTOT misses do not induce a complete order, as $C(t_i, lc_i)$ is non-convex. However, a conditional order can be induced when $r_i \leq r_j$, $lt_i \leq lt_j$, $i =_\delta j$, and $\mathcal{I}$ holds, under which sequencing i before j does not increase the combined delay and CTOT cost, encoded as (24).

$$\Phi \wedge \big(r_i \leq r_j \wedge i =_\delta j \wedge lt_i \leq lt_j \wedge \mathcal{I}\big) \wedge \neg\big(\sum_{\ell \in \mathcal{A}}(D_\ell + C_\ell) \leq \sum_{k \in \mathcal{A}}(D'_k + C'_k)\big) \tag{24}$$

4.4 Soundness of the Sequence Abstraction

Verification is performed on fixed-length sequences, whereas real instances admit sequences of arbitrary length. In (16), subsequences of aircraft preceding, following and in-between i and j are represented by single aircraft rather than arbitrary-length sequences; the validity of this representation must be justified.

In all cases, a subsequence is abstracted by a single aircraft in the model. This abstraction must preserve all effects relevant to rule correctness.

Binding Separations & Costs. Representative aircraft must capture the effects of subsequences on subsequent aircraft; thus, any binding separations on later aircraft must be equivalently imposed by the separations of the singular aircraft. This is discussed in the Prefix Subsequences explanation, below.

It is also necessary to capture the effects on the cost of the sequence being replaced, by the cost of this single aircraft replacing it. We ensure that the replacement aircraft is such that when the total cost of the original sequence would increase, so would the cost of the single aircraft. This is further justified in the Counterexamples explanation, below.

Establishing these two properties is sufficient, since all constraints are expressed in terms of aircraft timings (6)-(13), and the objective components (and therefore rule correctness) are also defined over such timings (4), (18).

Prefix Subsequences. For any aircraft k in sequence S, the takeoff time t_k is defined as $\max(r_k,\ \max_{x \in S_k}(t_x + \delta_{xk}))$. Although all predecessors impose separation constraints, t_k is only influenced by the maximum of these constraints; thus, the effect of S_k on t_k may be characterised solely by this maximal value.

Since δ-separations for a pair of aircraft are independent of the separations between other pairs, a symbolic ψ_1 can be defined to reproduce the binding constraints upon all later aircraft using (25). Since δ is arbitrary and symbolic, it covers every value induced by ψ_1, thus the binding effect of every prefix.

$$\delta_{\psi_1 k} = \max_{x \in S_k}(t_x + \delta_{xk}) - t_{\psi_1} \quad \forall k \in \{i,\ j,\ \psi_2,\ \psi_3\} \tag{25}$$

Counterexamples. If a counterexample exists for an arbitrary subsequence of aircraft after i or j, then swapping i and j must worsen at least one objective component. Since objectives are defined over individual aircraft, this implies the existence of at least one later aircraft ℓ whose contribution to some objective component worsens under the swap. We call this aircraft a witness.

Since any worsening sequence admits a witness, it suffices to represent only the aircraft whose takeoff time captures the worsening effect, rather than the entire subsequence. We now show how a single aircraft ψ_3 can accomplish this by considering the binding separations giving rise to the witness.

Binding Separations. Let ℓ be the witness aircraft that worsens under the swap. Since only i and j change position, any difference in t_ℓ can only be due to constraints originating from i or j. This can occur either directly, through a separation constraint between i or j and ℓ, or indirectly, where the effect of i or j propagates through intermediate aircraft before influencing t_ℓ.

Since separation parameters δ_{xy} are independent across aircraft pairs (separations between any two aircraft can be chosen without affecting others) and the solver is able to freely assign arbitrary non-negative values, the solver can freely choose δ-parameters to instantiate ψ_3 so as to reproduce the witness behaviour.

- If i or j influences t_ℓ directly, the solver can select ψ_3 with identical δ-parameters to ℓ, so that ψ_3 is subject to exactly the same binding constraints.
- If influence propagates through intermediate aircraft, it can select parameters of δ to impose the same constraint on ℓ as the full propagation chain.

In both cases, ψ_3 is able to reproduce the binding constraint for t_ℓ, and therefore attain the same takeoff time as ℓ. Since objectives depend only on these timing quantities[5], any worsening observed for ℓ under the swap is preserved by ψ_3, therefore the presence of a counterexample is maintained using only ψ_3.

Intermediate Segment. Aircraft between i and j may both influence a witness ℓ and act as a witness themselves. This is possible because separations from earlier aircraft, $\delta_{x\ell}$, are independent of separations to later aircraft, $\delta_{\ell y}$; thus, both effects on cost and on later aircraft can be represented by a single aircraft.

[5] Setting $(t_\ell,\ et_\ell,\ lt_\ell,\ ec_\ell,\ lc_\ell) := (t_\psi,\ et_\psi,\ lt_\psi,\ ec_\psi,\ lc_\psi)$ means that any objective component defined over these quantities evaluates identically for both ψ and ℓ.

5 Evaluation

We presented an SMT-based approach for verifying pruning rules in the Runway Sequencing Problem, showing that the conditional and complete order pruning rules can be encoded and checked automatically with the Z3 Theorem Prover.

All verification encodings were checked automatically by Z3. Across all verified pruning rules, solver runtimes were consistently low (sub-one-second). Our full correctness and non-vacuity results are summarised in Table 1. This shows that verifying RSP pruning rules using SMT is tractable; the effectiveness of the verified pruning rules (on problem instances) is discussed in [5].

Replicability. All SMT encodings used in this work are freely and publicly available online in a GitHub repository[6] as executable, annotated notebooks.

In this paper, we omit explicit formulations for conditional orders with unknown takeoff times for brevity; these are given in the accompanying repository. This repository includes the formalisation of the RSP constraints using the Z3 Theorem Prover, the encodings of pruning rules, and executable notebooks to reproduce both correctness/non-vacuity checks and show counterexamples.

Counterexamples & Synthesis. In our approach, refuting the correctness of a pruning rule constructs a concrete counterexample. Such instances can be used to guide the iterative refinement and debugging of pruning rules, laying the groundwork for future automated pruning rule synthesis methodologies.

Specifically, this suggests a direction towards counterexample-guided inductive synthesis (CEGIS), in which candidate rules would be iteratively synthesised and checked against a verifier, with counterexamples driving refinement. Such approaches have been successfully applied in the synthesis of programs [4].

Table 1. Summary of verified pruning rules.

Rule Type	Objective	Correct	Non-Vacuous
Complete Orders	*Makespan*	✓	✓
	Delay	✓	✓
	Time Windows	✓	✓
Conditional Orders (with known t_i / $\mathcal{I}$)	*Makespan*	✓	✓
	Delay & CTOT	✓	✓
	Time Windows	✓	✓
Conditional Orders (with unknown t_i)	*Makespan*	✓	✓
	Delay & CTOT	✓	✓
	Time Windows	✓	✓

[6] https://github.com/tobybenjaminclark/rsp-smt

Limitations. Our approach verifies a specific class of pruning rules, namely complete and conditional order rules over swapped sequences. Although this captures several published rules, broader classes of dominance arguments in [5] may require different abstractions or more expressive encodings.

Modifications to objective components which extend beyond the decidable fragments of SMT may render automated verification undecidable or less tractable, limiting the range of model variants in which the approach can be used.

While SMT provides automated, push-button verification, it offers limited insight into why the result is correct, placing a significant degree of trust in the solver implementation when it fails to find a counterexample.

Generalisation. While the case study focuses on runway sequencing, the approach is not domain-specific. Dominance relations and pruning rules are used elsewhere in exact optimisation, and are similarly amenable to verification.

Our approach recasts rule correctness as an invariant over the underlying model, decoupled from the optimisation procedure in which the rules are applied.

Conclusion. This paper presented an approach for verifying pruning rules using Z3, enabling automated correctness checking of complete and conditional orders.

The semantics of the RSP were formalised within decidable SMT theories, and verification over a finite sequence abstraction was justified as sufficient to guarantee correctness and non-vacuity for arbitrary-length sequences.

More broadly, this work illustrates how formal methods and optimisation can be hybridised to enable the automated verification of pruning rules, and suggests a natural future research direction towards their automated synthesis.

References

1. Allahverdi, A., Gupta, J.N., Aldowaisan, T.: A review of scheduling research involving setup considerations. Omega **27**(2), 219–239 (1999)
2. Allahverdi, A., Soroush, H.M.: The significance of reducing setup times/setup costs. European Journal of Operational Research **187**(3), 978–984 (2008)
3. Bjørner, N.S., McMillan, K.L., Rybalchenko, A.: Program verification as satisfiability modulo theories. SMT@ IJCAR **20**, 3–11 (2012)
4. David, C., Kroening, D.: Program synthesis: challenges and opportunities. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **375**(2104), 20150403 (2017)
5. De Maere, G., Atkin, J.A., Burke, E.K.: Pruning rules for optimal runway sequencing. Transportation Science **52**(4), 898–916 (2018)
6. Karapetyan, D., Gutin, G.: Solving the workflow satisfiability problem using general purpose solvers. IEEE Transactions on Dependable and Secure Computing **20**(6), 4474–4485 (2022)
7. de Moura, L.M., Bjørner, N.S.: Proofs and refutations, and Z3. In: LPAR Workshops. vol. 418, pp. 123–132. Doha, Qatar (2008)
8. Shirini, K., Aghdasi, H.S., Saeedvand, S.: A comprehensive survey on multiple-runway aircraft landing optimization problem. International Journal of Aeronautical and Space Sciences **25**(4), 1574–1602 (2024)

Iterated Local Search for the Trigger Arc Travelling Salesman Problem

Admir Kadriu and Kadri Sylejmani

Faculty of Electrical and Computer Engineering,
University of Prishtina, Prishtina, Kosovo
admir.kadriu@uni-pr.edu, kadri.sylejmani@uni-pr.edu

Abstract. The Trigger Arc Travelling Salesman Problem (TATSP) is a recently introduced variant of the TSP in which arc costs depend dynamically on the traversal order: traversing a *trigger* arc before a *target* arc may change the target's cost, with only the last trigger encountered being active. The problem arises in warehouse routing with compactable storage systems and has so far only been addressed by an integer linear programming formulation that fails to scale beyond approximately 25 nodes. We propose an Iterated Local Search (ILS) with a simulated annealing acceptance criterion, using first-improvement hill climbing over an Or-opt neighbourhood adapted for trigger-arc semantics via a delta cost evaluation scheme that traces trigger fan-out. Computational experiments on 55 benchmark instances from the MESS 2024 competition show that the approach achieves an average gap of 1.79% on the dense C1 instances (20–60 nodes) and 0.07% on the sparser C2 instances (18–142 nodes), establishing 5 new best-known solutions on the latter family while scaling well beyond the reach of exact methods.

Keywords: Travelling Salesman Problem · Iterated Local Search · Metaheuristics · Dynamic Arc Costs

1 Introduction and Problem Definition

The Travelling Salesman Problem (TSP) is one of the most extensively studied problems in combinatorial optimisation [1]. In the classical formulation, a salesman must visit a set of cities exactly once and return to the starting city, minimising the total travel cost. Numerous variants have been proposed to capture real-world complexities, including problems with traversal-dependent edge costs or structure [2,3], yet most assume that arc costs are fixed and independent of the solution structure. Cerrone et al. [4] recently introduced the *Trigger Arc Travelling Salesman Problem* (TATSP), a variant in which arc costs depend dynamically on the traversal order of arcs in the tour. The TATSP is motivated by the optimisation of picking routes in warehouses equipped with compactable storage systems, where mobile shelving units slide on rails: opening one aisle to retrieve items may close adjacent aisles, thereby changing the cost of accessing

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

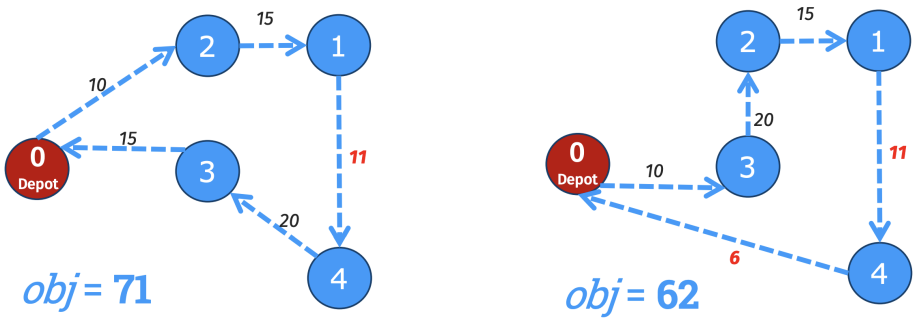


Fig. 1. Two tours on a 5-node TATSP instance with five trigger relations (Table 1). Triggered costs are shown in red. Left: tour 0-2-1-4-3-0, cost 71. Right: tour 0-3-2-1-4-0, cost 62 (optimal).

Table 1. Trigger relations for the example instance in Figure 1.

Relation	Trigger $\rightarrow$ Target	Triggered cost
r_1	$(0,2) \rightarrow (1,4)$	3
r_2	$(2,1) \rightarrow (1,4)$	11
r_3	$(2,1) \rightarrow (3,2)$	3
r_4	$(2,1) \rightarrow (4,0)$	6
r_5	$(3,0) \rightarrow (1,4)$	5

them later in the route. More generally, the model captures real-world scenarios in which the traversal of an arc by a vehicle can lead to modifications in the structure of the road graph [4].

We now state the problem formally, following [4]. Let $G = (N, A)$ be a directed graph with node set $N = \{0, 1, \ldots, n\}$, where node 0 is the depot, and arc set A. Each arc $a \in A$ has a base cost $c(a)$. For each arc a, there is a (possibly empty) set of *trigger relations* $R_a = \{(a_1, a) \mid a_1 \in A\}$. A relation $r = (a_i, a_j)$ carries a modified cost $c(r)$ and is said to be *active* in a tour T if both a_i and a_j belong to T and a_i is traversed before a_j, with no other relation $r' = (a_k, a_j) \in R_{a_j}$ such that a_k appears between a_i and a_j in T. In other words, only the *last* trigger arc encountered before a target arc determines its cost — a semantics we refer to as "last trigger wins." If no relation is active for an arc, it retains its base cost. The objective is to find a minimum-cost Hamiltonian cycle starting and ending at the depot.

Figure 1 illustrates the effect of trigger relations on a 5-node instance with five relations (Table 1). Tour $0 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 3 \rightarrow 0$ (left) traverses arc $(2,1)$ before arc $(1,4)$, activating r_2 and changing the cost of $(1,4)$ from 10 to 11; the earlier r_1 activated by $(0,2)$ is overridden, giving a total cost of 71. Tour $0 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 0$ (right) also activates r_2 on $(1,4)$ and additionally r_4 on $(4,0)$, reducing its cost from 10 to 6, yielding the optimal cost of 62.

The only existing exact approach is the integer linear programming (ILP) formulation proposed by Cerrone et al. [4]. Their computational experiments on complete directed graphs with up to 25 nodes and relation densities up to $16n^2$ show that the ILP solver struggles as instance size and relation density grow: at 25 nodes, MIP gaps reach 36% in Balanced scenarios and up to 42% in Decrease scenarios within a 1000-second time limit. Even at 20 nodes with high relation density, some instances cannot be solved to optimality. The authors explicitly note that "the implementation of heuristic techniques capable of quickly identifying good solutions to the problem could be interesting and useful."

In this paper, we propose an Iterated Local Search (ILS) [6] for the TATSP that combines a first-improvement hill climbing procedure using an Or-opt neighbourhood with a Simulated Annealing acceptance criterion [5]. We evaluate our method on the 55 benchmark instances from the MESS 2024 competition [7] and compare against the best-known solutions as well as the concurrent GRASP approach of Salvà Soler and de Lambertye [8]. Our ILS achieves an average gap of 1.79% on the dense C1 instances and 0.07% on the sparser C2 instances, establishing 5 new best-known solutions on the latter family while scaling to instances with up to 142 nodes.

2 Solution Approach

We propose an Iterated Local Search (ILS) [6] for the TATSP. The algorithm starts from a random initial tour, improves it via a first-improvement hill climbing procedure using an Or-opt neighbourhood adapted for trigger-arc semantics, and escapes local optima through segment-based perturbations combined with a simulated annealing (SA) [5] acceptance criterion. A key technical challenge is cost evaluation: because trigger relations link distant arcs, even a single-node move can change the effective cost of arcs far from the modification site. We address this with a delta evaluation scheme that traces trigger fan-out and falls back to full recomputation only when the affected region is large.

2.1 Cost Evaluation

In a standard TSP, moving a node affects at most a constant number of arcs, enabling $O(1)$ delta evaluation. In the TATSP, this property does not hold. Traversing an arc t before an arc a may change the cost of a to a triggered value c', and only the *last* such trigger encountered before a is active [4]. Consequently, reordering a single node can shift trigger–target orderings throughout the tour, potentially altering costs at positions far from the modification.

Full evaluation. Given a tour $\pi = (v_0, v_1, \ldots, v_n, v_0)$, the cost of each arc $a_i = (v_i, v_{i+1})$ is determined by walking backward through the previously traversed arcs to find the last trigger for a_i. If such a trigger exists, a_i takes its triggered cost; otherwise it retains its base cost. This yields an $O(n^2)$ worst-case procedure when trigger lookups are supported by a hash map (see Complexity below).

Delta evaluation. To avoid full recomputation on every candidate move, each solution caches its per-position arc costs and an arc-to-position mapping. Given a parent solution s and a neighbor s', delta evaluation proceeds as follows:

1. **Identify changed arcs.** Compare the tours of s and s' position by position to find directly changed arcs.
2. **Compute trigger fan-out.** For each changed arc that participates as a trigger in some relation (t, a, c'), mark the target arc a as affected (if it appears in the tour).
3. **Selective recomputation.** Copy the parent's cached costs and recompute only the affected positions using the backward-walk procedure.

When more than 50% of arcs are affected—either directly or through trigger fan-out—the method falls back to full evaluation, since selective recomputation offers no advantage in that regime.

Complexity. Full evaluation costs $O(n^2)$ in the worst case: each of the n arcs requires a backward walk of up to n positions, with each position checked against a precomputed hash map in $O(1)$. For delta evaluation, let c denote the number of tour positions where s and s' differ, and let $d_{\max}$ be the maximum trigger fan-out of any arc (i.e. the largest number of targets activated by a single trigger). Step 1 identifies the c changed positions in $O(n)$ time. Step 2 expands each changed position's fan-out, adding at most $c \cdot d_{\max}$ positions to the affected set; let a be the total number of affected positions. Step 3 recomputes each of the a affected positions via a backward walk costing up to $O(n)$, for a total of $O(a \cdot n)$. The overall delta evaluation cost is therefore $O(n + c \cdot d_{\max} + a \cdot n)$, with the fallback to full evaluation when $a > n/2$. On instances with moderate relation density, $a \ll n$, and delta evaluation yields a substantial speedup over full recomputation.

Handling infeasible solutions. Since the benchmark instances are not complete graphs, neighbourhood moves may produce tours that traverse arcs absent from the instance. Rather than constraining the neighbourhood operators to generate only feasible moves, we allow infeasible solutions during the search and penalise each missing arc with a cost of $5\,\bar{c}$, where $\bar{c}$ is the average base arc cost. This penalty is large enough to steer the search toward feasible solutions while preserving the simplicity and generality of the neighbourhood operators. Solution validity is verified after the search terminates.

2.2 Neighborhood Structure and Local Search

We considered three neighborhood operators, each generating candidate moves in randomized order. All operators preserve the depot at the first and last positions of the tour.

– **Or-opt** (k=1): Remove a single city from its current position and reinsert it at every other internal position.

Algorithm 1 Hill Climbing (First-Improvement)

Require: Solution s, instance I, neighborhood N
Ensure: Locally optimal solution s
  repeat
    improved $\leftarrow$ **false**
    for each neighbor s' generated by $N(s)$ in random order **do**
      if $f(s') < f(s)$ **then**
        $s \leftarrow s'$
        improved $\leftarrow$ **true**
        break {first improvement}
      end if
    end for
  until *improved* = **false**
  return s

- **Swap**: Exchange two cities at internal positions i and j.
- **Two-opt**: Reverse the sub-segment of the tour between internal positions i and j.

Each operator is implemented as a lazy generator that yields neighbors one at a time, with the cost of each neighbor computed via delta evaluation from the current solution. This avoids materializing the full neighborhood while preserving fast cost updates. A comparative analysis of the three operators (Section 3.2) shows that Or-opt consistently dominates, so all results use Or-opt as the sole neighbourhood.

The local search procedure is a first-improvement hill climbing (Algorithm 1). The algorithm scans candidates in random order and accepts the first improving move. When no improving move is found, the current solution is certified as a local optimum.

2.3 Iterated Local Search

The ILS framework [6] alternates between local search and perturbation to explore the space of local optima. Our implementation (Algorithm 2) constructs an initial tour by randomly permuting the internal nodes (with the depot fixed at positions 0 and $n+1$) and applies hill climbing to reach a first local optimum. In each subsequent iteration, the current solution is perturbed and then re-optimized by hill climbing. An SA acceptance criterion [5] governs whether the new solution replaces the current one, while the overall best solution is tracked separately.

Perturbation. A random contiguous segment of the tour is selected, with length equal to $\lfloor 0.2 \cdot n \rfloor$ internal nodes (minimum 2). One of three operators is then applied uniformly at random:

1. **Reverse**: reverse the order of nodes within the segment.

Algorithm 2 Iterated Local Search for the TATSP

Require: Instance I, parameters $maxIter$, ρ, T_0, α, $patience$
Ensure: Best solution s^*
  $s \leftarrow \text{RandomTour}(I)$
  $s \leftarrow \text{HillClimbing}(s, I)$ {Algorithm 1}
  $s^* \leftarrow s$; $T \leftarrow T_0$; $noImprove \leftarrow 0$
  for $i = 1$ **to** $maxIter$ **do**
    $s' \leftarrow \text{Perturb}(s, \rho)$ {reverse, relocate, or shuffle}
    $s' \leftarrow \text{HillClimbing}(s', I)$
    if $f(s') \le f(s)$ **or** $\text{rand}(0,1) < \exp\big(-(f(s') - f(s))/T\big)$ **then**
      $s \leftarrow s'$
      if $f(s') < f(s^*)$ **then**
        $s^* \leftarrow s'$; $noImprove \leftarrow 0$
      else
        $noImprove \leftarrow noImprove + 1$
      end if
    else
      $noImprove \leftarrow noImprove + 1$
    end if
    $T \leftarrow \alpha \cdot T$
    if $noImprove \ge patience$ **then**
      break
    end if
  end for
  return s^*

2. **Relocate**: remove the segment and reinsert it at a random internal position.
3. **Shuffle**: randomly permute the nodes within the segment.

This perturbation is strong enough to escape basins of attraction while preserving most of the tour structure.

Acceptance criterion. A candidate solution s' always replaces the current solution s if $f(s') \le f(s)$. Otherwise, it is accepted with probability $\exp\big(-(f(s') - f(s))/T\big)$, where T is the current temperature. The temperature follows a geometric cooling schedule $T \leftarrow \alpha \cdot T$ after each iteration, with initial temperature $T_0 = 50$ and cooling rate $\alpha = 0.99$.

Termination. The search terminates after a maximum of 5 000 iterations or after 200 consecutive iterations without improvement to the best-known solution, whichever comes first.

3 Computational Results

3.1 Instances and Experimental Setup

We evaluate our ILS on the 55 benchmark instances from the MESS 2024 competition [7], grouped into two families. The first family, **C1**, consists of 21 dense

instances (grf1–grf21): near-complete directed graphs with 20, 40, or 60 nodes (graph density 51–79%) and relation densities ranging from 2% to 65%, yielding between roughly 1 700 and 4.5 million trigger relations. Cost changes are balanced: triggers are approximately equally likely to increase or decrease an arc's cost.

The second family, **C2**, comprises 34 sparser instances (grf101–grf134) with 18 to 142 nodes. These graphs have lower density (8–29%) with more uniform relation densities (8–16%) and are organised into five sub-families of increasing size. Unlike C1, triggers here predominantly increase arc costs, penalising certain traversal orderings.

All experiments were run in parallel on a node of the Hyjnesha computing cluster at the University of Prishtina, equipped with two Intel Xeon Gold 6130 CPUs (64 logical cores total) and 128 GB of RAM. The solver is implemented in Python. We use the parameter setting described in Section 2.3: $I_{\max} = 5000$ iterations, perturbation strength 20%, initial temperature $T_0 = 50$, cooling rate $\alpha = 0.99$, and early termination after 200 consecutive non-improving iterations. Each instance was solved in 10 independent runs with different random seeds. Average running times range from under two minutes for the smallest 20-node C1 instances to several hours for the largest C2 instances with over 100 nodes.

3.2 Analysis of Neighbourhood Operators

To select the most effective neighbourhood operator, we ran the ILS with each of the three operators in isolation on a representative subset of instances from both families (5 runs per variant). Table 2 reports the results.

Table 2. Neighbourhood variant comparison: ILS with each operator in isolation (5 runs per variant). Dashes indicate no run completed within a reasonable time under the same iteration limit; superscripts denote the number of completed runs when fewer than 5. Best values per row in bold.

		Or-opt		Swap		2-opt	
Inst.	n	Best	Avg	Best	Avg	Best	Avg
grf1	20	**101.34**	102.44	104.62	106.46	107.00	112.63
grf8	40	**52.88**	53.59	56.25	61.24	58.55	72.85
grf17	60	**306.77**	312.39	323.95	328.00	345.80	352.05
grf101	18	**16.96**	16.97	17.12	17.15	17.11	17.19
grf105	26	**24.26**	24.33	24.38	24.88	24.38^{2}	25.76^{2}
grf112	50	**46.26**	46.42	–	–	–	–
grf117	52	**47.99**	48.29	48.33^{1}	48.33^{1}	–	–
grf129	72	**66.60**	67.19	–	–	–	–
grf131	100	$\mathbf{91.96}^{3}$	92.22^{3}	95.82^{1}	95.82^{1}	–	–
grf122	102	**93.80**	94.02	–	–	–	–
grf127	110	$\mathbf{100.96}^{4}$	101.27^{4}	–	–	–	–

Or-opt consistently produces the best solutions: on C1, the gap between Or-opt and 2-opt best values grows from 6% at 20 nodes to 13% at 60 nodes, while Swap is intermediate. On C2, the advantage is even more pronounced: for instances with $n \geq 50$, Swap and 2-opt rarely complete all five runs (and often complete none), whereas Or-opt scales to 110 nodes. Because trigger relations can propagate cost changes far from the modification site, the single-city reinsertion

of Or-opt explores a wider range of trigger–target reorderings per move than the segment-based operators, while keeping the per-iteration cost low. All results reported below therefore use Or-opt as the sole neighbourhood.

3.3 Comparison with State-of-the-Art Results

Table 3 reports the best and average objective values across all runs for every instance, together with the best-known solution value (BKS) from the MESS 2024 competition dashboard [7] and the percentage gap of our best solution from the BKS, defined as $\text{Gap} = 100 \cdot (f_{\text{best}} - f_{\text{BKS}})/f_{\text{BKS}}$. A negative gap indicates that our method found a new best-known solution.

Table 3. ILS results on C1 and C2 instances. Columns show the number of nodes n, the best-known solution (BKS) from the MESS 2024 competition, the best, average, and standard deviation (σ) of objective values across 10 independent runs, and the percentage gap of the best solution from the BKS.

Inst.	n	BKS	Best	Avg	σ	Gap%	Inst.	n	BKS	Best	Avg	σ	Gap%
C1 (grf1–grf21)							grf108	37	34.20	34.20	34.39	0.11	0.00
grf1	20	101.13	101.13	102.47	1.23	0.00	grf109	37	34.26	34.32	34.41	0.05	0.18
grf2	20	44.53	44.53	45.61	1.30	0.00	grf110	37	34.31	34.32	34.44	0.06	0.03
grf3	20	98.35	98.35	101.09	1.29	0.00	grf111	42	38.77	38.81	39.04	0.17	0.10
grf4	20	38.59	38.59	38.77	0.57	0.00	grf112	50	46.10	46.22	46.38	0.14	0.26
grf5	20	113.17	113.17	114.48	0.98	0.00	grf113	58	53.41	53.51	53.89	0.90	0.19
grf6	20	49.93	49.93	50.70	1.17	0.00	grf114	66	60.74	60.72	61.05	0.24	0.00
grf19	20	44.53	44.53	45.05	0.86	0.00	grf115	74	67.85	68.01	68.47	0.97	0.24
grf7	40	190.84	197.76	202.77	2.39	3.63	grf116	82	75.07	75.27	75.82	1.05	0.27
grf8	40	52.88	52.88	55.11	1.40	0.00	grf117	52	48.07	48.11	48.76	1.07	0.08
grf9	40	184.48	185.63	193.00	3.74	0.62	grf118	62	57.04	56.91	57.24	0.16	−0.23
grf10	40	55.88	56.30	59.07	1.86	0.75	grf119	72	66.23	66.16	66.83	1.03	0.00
grf11	40	204.90	210.23	217.87	2.89	2.60	grf120	82	75.38	75.31	75.98	1.12	0.00
grf12	40	62.36	64.28	70.39	4.70	3.08	grf121	92	84.27	84.42	84.85	0.27	0.18
grf20	40	52.88	52.88	55.56	1.92	0.00	grf122	102	93.37	93.61	94.22	1.08	0.26
grf13	60	282.27	292.85	298.64	3.60	3.75	grf123	62	57.04	56.89	57.67	0.42	−0.26
grf14	60	61.39	61.97	65.77	3.31	0.94	grf124	74	68.19	68.17	69.54	2.20	0.00
grf15	60	270.13	287.49	291.23	2.05	6.43	grf125	86	79.10	78.86	79.53	0.46	−0.30
grf16	60	68.83	71.94	76.48	2.74	4.52	grf126	98	90.00	90.09	90.99	1.47	0.10
grf17	60	293.36	302.97	312.24	4.50	3.28	grf127	110	100.76	100.74	102.78	3.81	−0.02
grf18	60	78.65	84.89	91.17	3.65	7.93	grf128	122	111.46	111.71	113.27	2.01	0.22
grf21	60	61.47	61.53	68.61	5.04	0.10	grf129	72	66.32	66.33	66.88	0.26	0.02
C1 avg. gap						1.79	grf130	86	79.01	79.52	82.74	5.09	0.65
C2 (grf101–grf134)							grf131	100	91.91	91.84	93.35	2.64	0.00
grf101	18	16.96	16.96	16.98	0.03	0.00	grf132	114	104.49	104.56	105.64	1.63	0.07
grf102	22	20.49	20.49	20.57	0.06	0.00	grf133	128	117.25	117.59	121.11	3.94	0.29
grf103	22	20.54	20.54	20.54	0.00	0.00	grf134	142	129.78	130.32	134.03	4.50	0.42
grf104	27	25.12	25.12	25.20	0.06	0.00							
grf105	26	24.26	24.26	24.33	0.06	0.00							
grf106	32	29.57	29.57	29.72	0.06	0.00							
grf107	37	34.26	34.22	34.34	0.05	−0.12							
C2 avg. gap						0.07	New BKS found						5/34

C1 instances. On the dense C1 instances, the ILS matches the best-known solution on all seven 20-node instances and two of the seven 40-node instances.

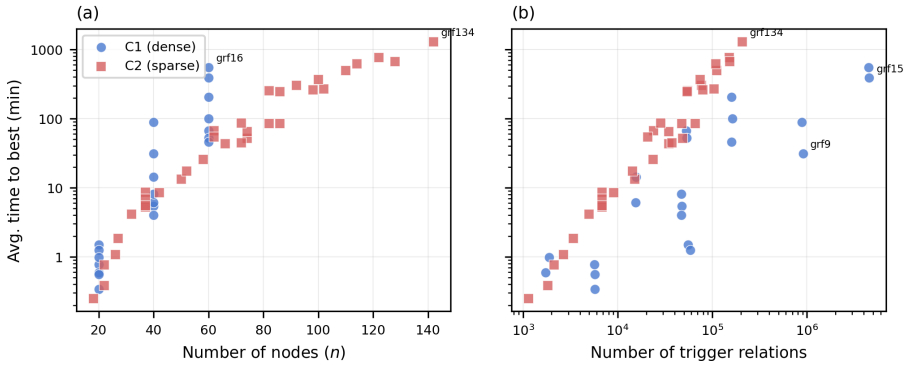


Fig. 2. Average time to best solution vs. (a) number of nodes and (b) number of trigger relations, coloured by instance family (log scale on all axes except (a) x-axis).

The average gap from the BKS across all 21 C1 instances is 1.79%. Performance degrades on larger and denser instances: the 60-node group has an average gap of roughly 3.9%, driven primarily by instances with high relation density (grf15, grf18) where the rugged search landscape with millions of trigger relations creates many local optima. Run-to-run variance is also noticeable on these instances, with the gap between the best and worst solution in 10 runs reaching up to 16% on the hardest 60-node instances.

C2 instances. On the sparser C2 family the ILS produces remarkably strong results, achieving an average gap of just 0.07% and establishing **5 new best-known solutions** out of 34 instances (indicated by negative gaps in Table 3). The gap between the best and average solution is typically below 1% for instances up to about 80 nodes and stays within 3% even at $n = 142$. Objective values scale nearly linearly with the number of nodes—roughly 0.92–0.95 per node—reflecting the regular structure of these sparser graphs. The new BKS are spread across all sub-families, including instances with up to 110 nodes (grf127), demonstrating that the ILS effectively exploits the sparser trigger structure of these instances.

Scalability. Figure 2 shows the average time to reach the best solution found in each run. Panel (a) reveals that, for a given node count, C1 instances require considerably more time than C2 instances due to their higher graph and relation density. Panel (b) shows that the number of trigger relations is a strong predictor of solving time across both families: instances with similar relation counts require comparable effort regardless of family membership. Average times range from under two minutes for the smallest instances to several hours for the largest C2 instances (grf134, 142 nodes) and the most relation-dense C1 instances (grf15, grf16, over four million relations).

Comparison with the GRASP of Salvà Soler and de Lambertye. The concurrent work by Salva Soler and de Lambertye [8] proposes a GRASP with a MIP-based construction heuristic and multi-neighbourhood local search, implemented in C++ with Gurobi. On C1, they report an average gap of 0.77% within a 60-second time limit, compared to our 1.79%. On C2, they report an average gap of 0.40%, whereas our ILS achieves 0.07% and finds 5 new best-known solutions. The gap on C1 is expected: their MIP-based construction provides high-quality starting solutions for the dense instances, and the C++ implementation with Gurobi is substantially faster than our Python solver. However, our ILS shows a clear advantage on C2, where the sparser trigger structure is better suited to the delta evaluation and iterative perturbation strategy.

Comparison with exact methods. A direct cost comparison with the ILP formulation of Cerrone et al. [4] is not possible, as their experiments use a different set of instances. However, their results demonstrate that the exact approach already encounters MIP gaps of 17–42% on complete graphs with only 25 nodes and 10 000 trigger relations, and hits the time limit of 1 000 seconds at 20 nodes with 6 400 relations. In contrast, our ILS handles instances with up to 142 nodes and millions of trigger relations. On the 20-node C1 instances, average running times are under two minutes; 40-node instances complete in minutes to about two hours; and even the largest C2 instances (110–142 nodes) complete within a day on a single core. This confirms that metaheuristic approaches are essential for tackling practically-sized TATSP instances.

4 Conclusion

We presented an Iterated Local Search for the Trigger Arc Travelling Salesman Problem [4], a TSP variant in which traversing certain arcs dynamically alters the costs of other arcs. Our method combines a simulated annealing acceptance criterion with a first-improvement hill climbing phase using an Or-opt neighbourhood—selected after preliminary experiments showed it consistently outperformed configurations including swap and 2-opt—augmented with a delta cost evaluation scheme that exploits trigger-relation caching. Computational experiments on the 55 benchmark instances from the MESS 2024 competition [7] demonstrate that the approach achieves an average gap of 1.79% on the dense C1 instances and just 0.07% on the sparser C2 family, establishing 5 new best-known solutions on the latter. The ILS scales to instances with up to 142 nodes, well beyond the practical reach of existing exact methods, which are limited to roughly 25 nodes. Compared with the concurrent GRASP approach of Salva Soler and de Lambertye [8], our method is particularly effective on C2, while the GRASP shows stronger performance on the dense C1 instances thanks to its MIP-based construction heuristic.

Several directions for future work appear promising. Replacing the random initial solution with a construction heuristic—such as a nearest-neighbour or a trigger-aware greedy procedure—could provide higher-quality starting points

and reduce the number of ILS iterations needed to reach good solutions. An adaptive perturbation mechanism that learns which perturbation strategy is most effective for a given instance structure would further improve robustness. Parallel multi-start search could exploit modern multi-core architectures to explore the solution space more broadly. Finally, hybrid matheuristic approaches that solve subproblems exactly via integer linear programming while using metaheuristic search at the global level offer a compelling avenue, as does evaluation on larger instances and real-world warehouse routing data.

References

1. Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.J.: The Traveling Salesman Problem: A Computational Study. Princeton University Press (2011). https://doi.org/10.1515/9781400841103
2. Bossek, J., Casel, K., Kerschke, P., Neumann, F.: The node weight dependent traveling salesperson problem: Approximation algorithms and randomized search heuristics. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference. pp. 1286–1294. ACM (2020). https://doi.org/10.1145/3377930.3390243
3. Carmesin, S., Woller, D., Parker, D., Kulich, M., Mansouri, M.: The Hamiltonian cycle and travelling salesperson problems with traversal-dependent edge deletion. Journal of Computational Science **74**, 102156 (2023). https://doi.org/10.1016/j.jocs.2023.102156
4. Cerrone, C., Truvolo, M., Dragone, R., Battaglia, R.: The trigger arc TSP: Optimise the picking process in warehouses with compactable storage systems. In: Data Science and Intelligent Systems. Lecture Notes in Computer Science, vol. 14778, pp. 279–289. Springer Nature Switzerland, Cham (2025). https://doi.org/10.1007/978-3-031-78238-1_26
5. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983). https://doi.org/10.1126/science.220.4598.671
6. Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search: Framework and applications. In: Handbook of Metaheuristics, pp. 129–168. Springer, 3rd edn. (2019). https://doi.org/10.1007/978-3-319-91086-4_5
7. Metaheuristics Summer School: MESS 2024 Competition Dashboard. https://fourclicks.eu/fck/mess2024/frontend/#/home/dashboard (2024), accessed: 2026-03-14
8. Salvà Soler, J., de Lambertye, G.: A fast GRASP metaheuristic for the trigger arc TSP with MIP-based construction and multi-neighborhood local search. arXiv preprint arXiv:2508.08477 (2025). https://doi.org/10.48550/arXiv.2508.08477

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

An Iterative Model Reduction Method and a Constraint Programming Formulation for the Train Dispatching Problem

László Kálmán Trautsch[0000−0002−1589−3265] and
Bence Kovari[0000−0003−1555−640X]

Department of Automation and Applied Informatics, Budapest University of Technology and Economics, Műegyetem rkp. 3., Budapest, H-1111, Hungary
trautschl@edu.bme.hu, kovari@aut.bme.hu

Abstract. This paper addresses the train dispatching problem, which involves rescheduling and rerouting delayed trains on a shared railway network to minimize total delay and avoid resource conflicts. Dispatching optimization can reduce overall lateness, increase network throughput, and improve the utilization of railway infrastructure. We propose an iterative model reduction method that reduces the complexity of problem instances while ensuring that no optimal solution is eliminated from the search space. A series of model transformations are introduced that tighten operation start-time bounds, merge compatible operations, prune dominated subpaths, remove redundant resource requirements, and linearize train operation graphs. The proposed reduction method can be applied either as a preprocessing step or dynamically within other solution methods. In this paper, we apply the reduction as preprocessing and propose a compact constraint programming formulation based on the reformulated model. The solution method is evaluated on the DISPLIB 2025 benchmark library, which consists of instances derived from real-world use cases. The computational results show that the proposed model reduction method can substantially simplify large problem instances and improve performance. The complete solution method was able to find high-quality feasible solutions for most benchmark instances and achieved the fourth-highest overall score in the final phase of the DISPLIB 2025 competition.

Keywords: Train Dispatching · Model Reduction · Constraint Programming

1 Introduction

Train dispatching problems arise frequently in railway systems, particularly when unexpected delays or disruptions require rapid routing and scheduling adjustments to minimize overall delay and maintain service quality. Effective dispatching decisions are essential for reliable and efficient railway traffic management, especially in dense networks where even minor disruptions can propagate quickly.

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Consequently, there is considerable interest in both academia and industry in developing robust and efficient methods for real-time train dispatching.

Accordingly, train dispatching and railway rescheduling have been studied for decades, as reflected already in the survey of Cordeau et al. [1]. In the real-time setting, a major line of work models dispatching as a combined routing and scheduling problem under infrastructure conflicts. D'Ariano et al. [2] proposed a branch-and-bound method for train scheduling in a railway network, and D'Ariano et al. [3] further considered conflict resolution together with train speed coordination for real-time timetable perturbations.

Subsequent work addressed several important variants of the problem. Mannino and Mascis [4] studied optimal real-time traffic control in metro stations, while Corman et al. [5] compared centralized and distributed approaches for rescheduling trains in multiple dispatching areas. More recently, optimization-based methods have been complemented by learning-based approaches: Agasucci et al. [6] applied deep reinforcement learning to the train dispatching problem. Exact and hybrid methods also show continuing progress, for example through column generation for real-time dispatching [7], constraint-based in-station dispatching [8], and hybrid optimization for complex railway stations [9].

Several studies have addressed the computational difficulty of railway scheduling and rescheduling by reducing the set of routing or sequencing decisions before or during optimization. Zwaneveld et al. [10] formulated train routing through stations as a node packing problem and used preprocessing to remove superfluous routing alternatives. Vansteenwegen et al. [11] used dominance-based preprocessing to discard train route alternatives in a maintenance-rescheduling setting. Other approaches reduce the problem by decomposition, for example by separating line and station decisions in real-time dispatching [12]. Decomposition principles for large-scale railway scheduling are reviewed by Leutwiler and Corman [13].

In this paper, we address the train dispatching problem by proposing an iterative model reduction method that can significantly reduce problem complexity. The method combines several optimality-preserving transformations, including start-time bound tightening, operation merging, dominated-subpath pruning, redundant-resource deletion, and operation graph linearization. Rather than reducing only the set of admissible routes or decomposing the problem into smaller subproblems, the proposed method applies iterative reductions directly to the operation-graph representation of the problem, deriving a reduced model for optimization. Based on the reformulated model, we also develop a compact constraint programming formulation. We evaluate the proposed method on the publicly available instances of the DISPLIB 2025 library [14], which represent challenging real-world dispatching scenarios.

The remainder of this paper is organized as follows. Section 2 provides an overview of the train dispatching problem addressed in this paper. Section 3 describes the proposed iterative model reduction method, and Section 4 introduces the constraint programming formulation based on the reformulated model. Section 5 presents the computational results on the DISPLIB 2025 benchmark

library. Finally, Section 6 concludes the paper and outlines directions for future work.

2 Problem Definition

The train dispatching problem, as defined by the DISPLIB 2025 benchmark library [14], involves routing and scheduling a set of trains, each represented by a directed acyclic graph called an operation graph. The nodes of this graph represent train operations, while the edges represent immediate precedence relations between operations. For each train, alternative routes correspond to alternative entry-to-exit paths in its operation graph. Operations represent possible states or activities of a train, typically involving the occupation of one or more infrastructure resources. Resources represent infrastructure elements that can be occupied by at most one train at a time, such as physical track sections or block sections imposed by railway safety regulations. Each operation can have a specified minimum duration, lower and upper bounds on allowable start times, and may require exclusive access to one or more resources. The notation used in this paper and the input data are presented in Table 1.

Table 1. Notation for the input data

Symbol	Definition
I	Set of trains.
O	Set of all operations.
$O_i \subset O$	Set of operations of train i. $\{O_i\}_{i \in I}$ forms a partition of O.
$B, E \subset O$	Sets of entry and exit operations, respectively.
$S_o, P_o \subset O$	Sets of immediate successors and predecessors of operation o, respectively.
$\alpha_o, \beta_o \in \mathbb{Z}_{\geq 0}$	Lower and upper bounds on the start time of operation o, respectively.
$\delta_o \in \mathbb{Z}_{\geq 0}$	Minimum duration of operation o.
R	Set of resources.
$R_o \subseteq R$	Set of the resources required by operation o.
$\lambda_{r,o} \in \mathbb{Z}_{\geq 0}$	Release time of resource r required by operation o, i.e., the additional time that must elapse after o has ended before r can be used by an operation of another train.
C	Set of cost components.
$K_o \subset O$	Set of operations potentially conflicting with operation o, i.e., operations that may require a common resource with o.
$\mu_c \in \mathbb{Z}_{\geq 0}$	Delay threshold of cost component c.
$\omega_c, \nu_c \in \mathbb{Z}_{\geq 0}$	Linear and constant cost weights of cost component c, respectively.
$\phi : C \to O$	$\phi(c)$ is the operation associated with cost component c.

An operation graph contains exactly one entry operation, which has no incoming edges, and exactly one exit operation, which has no outgoing edges. A feasible solution to a problem instance is a chronologically ordered sequence of start events for selected operations across all trains. This global sequence must

respect the structure of the operation graphs, forming a complete path from entry to exit operations for each train, while also satisfying the start-time and minimum-duration bounds of all selected operations.

In addition, resource requirements impose further restrictions: no two trains may simultaneously occupy the same resource. If operation o occupies resource r, then an operation of another train can use r only after o has ended and the additional release time $\lambda_{r,o}$ has elapsed. An operation is considered to end when the next selected operation of the same train starts.

Feasible solutions are evaluated based on an objective function that minimizes delays and operational costs. The input data contains a set of cost components, each specifying a delay threshold for an associated operation, as well as linear and constant cost weights associated with the delay. A cost component is associated with one operation, while each operation may have any number of cost components. A cost component c is triggered if the associated operation $\phi(c)$ starts at or after its delay threshold μ_c. In that case, it contributes a constant cost ν_c and a linear cost proportional to the delay beyond the threshold, with weight ω_c. The objective function is defined in (1), where $t_{\phi(c)}$ is the assigned start time of the operation associated with cost component c, and H denotes the Heaviside step function, which takes the value 0 for negative arguments and 1 otherwise.

$$\min \sum_{c \in C} (\omega_c \cdot \max(0, t_{\phi(c)} - \mu_c) + \nu_c \cdot \mathrm{H}(t_{\phi(c)} - \mu_c)) \tag{1}$$

3 Model Reduction

To reduce problem size, we propose an iterative model reduction method that tightens lower bounds, merges operations, prunes dominated subpaths, and removes redundant resource requirements. We apply this method as a preprocessing step, but the transformations are designed so that they can also be applied efficiently during the execution of solver algorithms. To further reduce problem complexity, we also reformulate the model by linearizing the operation graphs and introducing path variables. This section presents these model reduction steps. For brevity, we omit the proofs that these transformations neither remove nor introduce any optimal solutions.

First, the start-time lower bound of each operation o is tightened by forward propagation, based on the earliest feasible end time among its immediate predecessors: $(\alpha_o \leftarrow \max(\alpha_o, \min_{o' \in P_o}(\alpha_{o'} + \delta_{o'})))$

3.1 Operation Merging

Certain operations with identical resource requirements can be merged into a single operation. The original indices of the merged operations are stored, allowing the solution to be reconstructed.

Sequential Operation Merging can be applied to a pair of operations o and o' that require identical resources ($R_o = R_{o'}$) if $S_o = \{o'\}$, $P_{o'} = \{o\}$ and o

does not have any resource requirements that can extend beyond those of operation o' ($\forall r \in R_o : \lambda_{r,o} \leq \lambda_{r,o'} + \delta_{o'}$). The delay threshold of any cost component c associated with o' is reduced by the minimum duration of o ($\mu_c \leftarrow \mu_c - \delta_o$), and the associated operation of c is updated to o ($\phi(c) \leftarrow o$). The minimum duration of operation o is then increased by the minimum duration of o' ($\delta_o \leftarrow \delta_o + \delta_{o'}$). For every resource r required by operation o, the associated release time is updated to that of o' ($\lambda_{r,o} \leftarrow \lambda_{r,o'}$). If operation o' is an exit operation of a train, o becomes the new exit ($E \leftarrow (E \cup \{o\}) \setminus \{o'\}$). Operation o acquires all immediate successors of o' and o' is removed from the immediate successors of o. ($S_o \leftarrow (S_o \cup S_{o'}) \setminus \{o'\}$, $P_{o'} \leftarrow \emptyset$, $\forall o'^{+} \in S_{o'} : P_{o'^{+}} \leftarrow (P_{o'^{+}} \cup \{o\}) \setminus \{o'\}$, $S_{o'} \leftarrow \emptyset$)

Parallel Operation Merging can be applied to a pair of operations o and o' that satisfy $P_o = P_{o'}$ or $S_o = S_{o'}$, $\alpha_o = \alpha_{o'}$, $\delta_o = \delta_{o'}$, $R_o = R_{o'}$, and $\forall r \in R_o : \lambda_{r,o} = \lambda_{r,o'}$, and for which each cost component associated with o has a corresponding cost component associated with o' whose parameters match ($\forall c \in C : (\phi(c) = o) \Rightarrow (\exists c' \in C : (\phi(c') = o') \wedge (\mu_c = \mu_{c'}) \wedge (\omega_c = \omega_{c'}) \wedge (\nu_c = \nu_{c'}))$). Similarly, each cost component associated with o' should also have a corresponding cost component associated with o. Since this merging requires the two operations to have similar properties, merging o' into o only requires updating the predecessor and successor relations. Operation o acquires all immediate predecessors and successors of o' ($P_o \leftarrow P_o \cup P_{o'}$, $S_o \leftarrow S_o \cup S_{o'}$, $P_{o'} \leftarrow \emptyset$, $S_{o'} \leftarrow \emptyset$), and the sets of immediate predecessors and successors of the adjacent operations are updated accordingly. ($\forall o^{-} \in P_o : S_{o^{-}} \leftarrow (S_{o^{-}} \cup \{o\}) \setminus \{o'\}$, $\forall o^{+} \in S_o : P_{o^{+}} \leftarrow (P_{o^{+}} \cup \{o\}) \setminus \{o'\}$)

3.2 Subpath Pruning

Certain subpaths in the operation graphs may be dominated by alternative subpaths and can therefore be eliminated. To reduce computational complexity, we examine only linear sequences of operations, meaning sequences in which every operation, except the first and the last, has exactly one immediate successor, or every operation, except the first and the last, has exactly one immediate predecessor.

Let $\pi = (o_1, \ldots, o_k)$ and $\pi' = (o'_1, \ldots, o'_\ell)$ be two sequences of operations corresponding to subpaths of an operation graph, with $o_1 = o'_1$, $o_k = o'_\ell$, and $k, \ell \geq 2$. We say that π dominates π' if there exists an integer $m \geq 1$ and decompositions of π and π' into m consecutive segments, $\pi = \pi^{(1)} \cdots \pi^{(m)}$ and $\pi' = \pi'^{(1)} \cdots \pi'^{(m)}$, such that for each $j = 1, \ldots, m$, the segment $\pi^{(j)}$ dominates the segment $\pi'^{(j)}$. Here, a segment $\pi^{(j)} = (o_1^{(j)}, \ldots, o_{k_j}^{(j)})$ dominates segment $\pi'^{(j)} = (o_1'^{(j)}, \ldots, o_{\ell_j}'^{(j)})$ if the following conditions hold:

- The total minimum duration of segment $\pi^{(j)}$ is less than or equal to that of segment $\pi'^{(j)}$. ($\sum_{i=1}^{k_j} \delta_{o_i^{(j)}} \leq \sum_{i=1}^{\ell_j} \delta_{o_i'^{(j)}}$)
- Segment $\pi^{(j)}$ does not need to start later than segment $\pi'^{(j)}$. ($\alpha_{o_1^{(j)}} \leq \alpha_{o_1'^{(j)}}$)

- Segment $\pi^{(j)}$ does not contain operations that require additional resources, or the same resources with larger release times, than any operation in segment $\pi'^{(j)}$. $(\forall o \in \pi^{(j)}, \forall o' \in \pi'^{(j)}, \forall r \in R_o : (r \in R_{o'}) \wedge (\lambda_{r,o} \leq \lambda_{r,o'}))$
- Operations $o_1^{(j)}$ and $o_1'^{(j)}$ are the only operations in segments $\pi^{(j)}$ and $\pi'^{(j)}$ that may be associated with cost components. $(\forall c \in C : \phi(c) \in \pi^{(j)} \cup \pi'^{(j)} \Rightarrow \phi(c) \in \{o_1^{(j)}, o_1'^{(j)}\})$
- Each cost component associated with operation $o_1^{(j)}$ has a corresponding cost component associated with operation $o_1'^{(j)}$ whose parameters are at least as restrictive. $(\forall c \in C : (\phi(c) = o_1^{(j)}) \Rightarrow$ $(\exists c' \in C : (\phi(c') = o_1'^{(j)}) \wedge (\mu_c \geq \mu_{c'}) \wedge (\omega_c \leq \omega_{c'}) \wedge (\nu_c \leq \nu_{c'})))$

If a sequence of operations π' is dominated by another sequence π, and every operation in π', except the first and the last, has exactly one immediate successor $(\forall n \in \{2, \ldots, \ell-1\} : S_{o'_n} = \{o'_{n+1}\})$, then the first internal edge of π' is removed from the graph, since any path containing that edge must follow a dominated subpath $(S_{o'_1} \leftarrow S_{o'_1} \setminus \{o'_2\}, \quad P_{o'_2} \leftarrow P_{o'_2} \setminus \{o'_1\})$. The reverse case is handled analogously: if every internal operation has exactly one immediate predecessor, the last internal edge of π' is removed.

After the pruning steps, any operation o that becomes unreachable ($S_o = \emptyset$ or $P_o = \emptyset$) is removed from the graph, as well as from the successor and predecessor sets of all other operations. $(\forall o' \in O : P_{o'} \leftarrow P_{o'} \setminus \{o\}, S_{o'} \leftarrow S_{o'} \setminus \{o\})$

3.3 Resource Deletion

A resource r that is required only by the operations of a single train $(\exists i \in I : \forall o \in O \setminus O_i : r \notin R_o)$ cannot induce conflicts and is therefore removed. If a resource r is always required together with another resource r' $(\forall o \in O : r \in R_o \Rightarrow r' \in R_o)$, and for every operation o that requires r we have $\lambda_{r,o} \leq \lambda_{r',o}$, then r is redundant and therefore removed. $(\forall o \in O : R_o \leftarrow R_o \setminus \{r\})$

3.4 Iterative Model Reduction

The previously presented model reduction transformations are applied iteratively and hierarchically. After each merging transformation, the affected operations are re-examined for further merging and possible tightening of start-time lower bounds. Similarly, whenever a dominated path is pruned or an unreachable operation is removed, the affected part of the graph is re-examined for newly dominated paths, newly unreachable operations, further merging, and further lower-bound tightening. Resource deletions may trigger further resource deletions, path pruning, operation merging, and lower-bound tightening.

As part of preprocessing, four topological traversals are performed on each operation graph. The first traversal performs lower-bound tightening, the second applies all eligible merging transformations, the third prunes dominated paths and removes unreachable operations, and the fourth deletes redundant resources. During each traversal, every reduction may immediately trigger further applicable reductions, and this effect may propagate through the entire graph.

3.5 Path Variable Assignment

In this step, the possible paths in the operation graph of each train are decomposed into sets of subpaths, and a unique path variable is assigned to each set to indicate whether the subpaths in that set are selected in the solution. The procedure begins by assigning a path variable to every operation and every possible succession between operations. An operation is selected if and only if it has exactly one selected incoming edge and one selected outgoing edge, except for entry and exit operations, which are always selected. Based on this, equality relationships between sums of variables are recorded to identify variables that must always be equal, and such variables are then replaced by a single path variable. As a result, the entry and exit operations of the graph are assigned the same path variable, and a single variable can indicate the selection of multiple operations and successions across multiple subpaths rather than only one operation or succession.

3.6 Operation Graph Linearization

Finally, to reduce problem complexity, the operation graph of each train is linearized into a single chain of aggregated operations. Each aggregated operation a is assigned a set G_a of original operations associated with it, and a set S_a^{agg} containing at most one immediate successor aggregated operation. Starting from the entry operation, the graph is processed iteratively by maintaining the set of currently reachable but not yet processed operations. In each step, operations from this set whose predecessors have all already been processed are grouped into a new aggregated operation, which becomes the immediate successor of the previous aggregated operation, and the set of reachable operations is updated.

After the operation graph linearization, the path variables ensure that, for each train, a single valid path of original operations is selected in the solution, and that the constraints defined for the selected original operations are enforced on the aggregated operations containing them. In this way, the subsequent model formulation requires fewer variables and constraints, and these become more strongly linked, while the solution space remains unchanged.

4 Constraint Programming Model

After applying the model reduction transformations, we obtain a reformulated model with linearized operation graphs, which form the basis of the proposed compact constraint programming model. This linearization allows operation conflict constraints to be formulated based on the start times of successor aggregated operations, thereby reducing the number of decision variables and linking the corresponding constraints more directly. In this section, we introduce the decision variables and constraints of the CP formulation. Table 2 summarizes the additional notation and decision variables used in the formulation. Throughout this section, the superscript "agg" denotes the aggregated-operation counterpart of notation previously defined for operations; in particular, B^{agg}, E^{agg}, P_a^{agg}, and S_a^{agg} denote the corresponding sets for aggregated operations.

Table 2. Notation used in the CP formulation

Symbol	Definition
A	Set of aggregated operations.
$G_a \subset O$	Set of original operations associated with aggregated operation a. $\{G_a\}_{a \in A}$ forms a partition of O.
$\eta_{o,o'} \in \mathbb{Z}_{\geq 0}$	The maximum resource release time required between the end of operation o and the start of operation $o' \in K_o$, if o precedes o'.
$\psi : C \to A$	$\psi(c)$ is the aggregated operation associated with cost component c.
X	Set of path decision variables. Each variable corresponds to one or more subpaths in the operation graph of a train. For $x \in X$, x is 1 if all subpaths associated with x are traversed by the train, 0 otherwise.
$t_a \in \mathbb{Z}_{\geq 0}$	Start time decision variable of aggregated operation a.
$q_{a,a'} \in \{0,1\}$	Binary decision variable equal to 1 if aggregated operation a must finish before aggregated operation a' starts, and 0 otherwise.
$y_c \in \{0,1\}$	Binary decision variable equal to 1 if cost component c is triggered, and 0 otherwise.
$z_c \in \mathbb{Z}_{\geq 0}$	Decision variable equal to the linear cost induced by cost component c when it is triggered.
$\chi^{\text{op}} : O \to X$	$\chi^{\text{op}}(o)$ is the path variable associated with operation o.
$\chi^{\text{suc}} : O \times O \to X$	$\chi^{\text{suc}}(o, o')$ is the path variable associated with the succession between operation o and o', where $o \in O$ and $o' \in S_o$.

4.1 Constraints and Objective

Correct operation assignments are enforced through constraints on the path variables. Constraints 2, 3, and 4 ensure path consistency and that exactly one path is selected from the entry operation to the exit operation of each train.

$$\chi^{\text{op}}(o) = 1 \quad \forall o \in B \tag{2}$$

$$\chi^{\text{op}}(o) = \sum_{o^- \in P_o} \chi^{\text{suc}}(o^-, o) \quad \forall o \in O \setminus B \tag{3}$$

$$\chi^{\text{op}}(o) = \sum_{o^+ \in S_o} \chi^{\text{suc}}(o, o^+) \quad \forall o \in O \setminus E \tag{4}$$

The correct start times and minimum durations of operations on selected subpaths are enforced by Constraints 5 and 6, while Constraints 7 ensure that aggregated operations containing no selected operations have zero duration.

$$\chi^{\text{op}}(o) \Rightarrow \alpha_o \leq t_a \leq \beta_o \quad \forall a \in A,\, o \in G_a \tag{5}$$

$$t_a \geq t_{a^-} + \sum_{o \in G_{a^-}} \delta_o \cdot \chi^{\text{op}}(o) \quad \forall a \in A \setminus B^{\text{agg}},\, a^- \in P_a^{\text{agg}} \tag{6}$$

$$\left(\neg \bigvee_{o \in G_a} \chi^{\text{op}}(o)\right) \Rightarrow t_a = t_{a^+} \quad \forall a \in A \setminus E^{\text{agg}},\, a^+ \in S_a^{\text{agg}} \tag{7}$$

Constraints 8 enforce a precedence decision between each pair of aggregated operations that contain potentially conflicting selected operations.

$$(\chi^{\text{op}}(o) \wedge \chi^{\text{op}}(o')) \Rightarrow q_{a,a'} + q_{a',a} = 1 \quad \forall a, a' \in A,\, o \in G_a,\, o' \in G_{a'} \cap K_o \tag{8}$$

Constraints 9 enforce that if an aggregated operation a precedes another aggregated operation a', then a' cannot start before the end time of a plus the maximum release time associated with potentially conflicting pairs of selected operations in a and a'. The end time of a is the start time of its successor aggregated operation a^+.

$$\begin{aligned}(q_{a,a'} \wedge \chi^{\text{op}}(o) \wedge \chi^{\text{op}}(o')) \Rightarrow t_{a'} \geq t_{a^+} + \eta_{o,o'} \\ \forall a \in A \setminus E^{\text{agg}},\, a' \in A,\, a^+ \in S_a^{\text{agg}},\, o \in G_a,\, o' \in G_{a'} \cap K_o\end{aligned} \tag{9}$$

No aggregated operation may precede the entry aggregated operation of a train, and the exit aggregated operation of a train may not precede any other aggregated operation. This is enforced by Constraints 10.

$$q_{a,a'} = 0 \quad \forall (a, a') \in (E^{\text{agg}} \times A) \cup (A \times B^{\text{agg}}) \tag{10}$$

Constraints 11 ensure that if the operation associated with a cost component c is selected, then either the start time of the associated aggregated operation remains below the delay threshold or the variable y_c takes the value 1, indicating that cost component c is triggered. Constraints 12 enforce that if c is triggered, then the corresponding cost variable z_c is equal to the weighted delay beyond the threshold.

$$(\neg y_c \wedge \chi^{\text{op}}(\phi(c))) \Rightarrow t_{\psi(c)} < \mu_c \quad \forall c \in C \tag{11}$$

$$y_c \Rightarrow z_c = \omega_c \cdot (t_{\psi(c)} - \mu_c) \quad \forall c \in C \tag{12}$$

The objective of the problem is to minimize the total cost induced by the cost components. Each triggered cost component c contributes a constant cost ν_c and a linear delay cost given by the decision variable z_c, as defined in (13).

$$\min \sum_{c \in C} (\nu_c \cdot y_c + z_c) \tag{13}$$

4.2 Deadlock Prevention

In our CP model, certain configurations can lead to circular-wait deadlocks that are not excluded by the constraints presented above. Specifically, if certain resource release times are zero, the model may allow a set of operations to start simultaneously even if each of these operations depends on another to release a required resource first. This creates a cyclic dependency that is feasible in the CP model but contradicts the problem definition.

To eliminate the possibility of such deadlocks, every zero release time is replaced by a small positive value $\epsilon < 1$, thereby enforcing a strict ordering between successive uses of the same resource, without introducing additional variables and constraints that would substantially increase the complexity of the model. Since all original constants and variable domains are integer-valued, a suitable value of ϵ can always be chosen so that the set of intended optimal solutions is preserved. Specifically, we set $\epsilon = 0.001$ and scale all constants and variable domains by $1/\epsilon$ so that they remain integer-valued. We also modify the constraints related to the cost components to ensure that these small delays do not affect the objective value.

We also employ auxiliary constraints that directly rule out two-train deadlock configurations and infeasible precedence relations between two trains. In particular, if $q_{a,a'} = 1$ for some aggregated operations a and a', then the opposite ordering cannot be assigned to the corresponding earlier and later aggregated operations of the two trains, as well as for the immediate predecessors and successors of a and a', so the relevant $q_{\hat{a}',\hat{a}}$ variables are set to 0. These constraints are not required for correctness, but can improve computational performance in practice. The constraints can be applied only if the operation graph of each train has been linearized.

5 Experimental Results

5.1 Experimental Setup

The algorithm was tested on the instances of the publicly available benchmark library developed for the DISPLIB 2025 competition [14]. The library contains 112 different real-world problem instances of varying difficulty, ranging from 4 to 505 trains, 113 to 52,741 operations, and 17 to 50,910,450 potential conflicts. The instances are divided into 13 categories, based on how they were produced.

The solution method was implemented in Python, using the CP-SAT solver [15] for the constraint programming model. Experiments were run on a machine with an AMD Ryzen 9 9950X processor (4.3 GHz base clock, up to 5.7 GHz boost) and 192 GB of RAM. In accordance with the DISPLIB 2025 competition rules [14], we limited each run to 8 CPU threads, 32 GB of RAM, and a 10-minute wall-time per instance. For each tested configuration, a single run of the solution method was performed on each problem instance of the dataset.

5.2 Results

The results of the solution method, aggregated by instance category, are presented in Table 3. To assess the effect of the iterative model reduction method, we compare the results obtained with a CP formulation based on the original model to those obtained with the presented CP formulation based on the preprocessed model. The comparison shows that the iterative model reduction method substantially improves the computational performance of the solution method, most notably in the number of feasible solutions found. Without preprocessing, feasible solutions are obtained for only 55 of the 112 problem instances, whereas with preprocessing the solution method finds feasible solutions for 100 instances. The table also reports the average solution time, with a value of 600 s assigned to instances that were not solved to optimality within the time limit or for which the solution method exhausted the available memory. This occurred frequently without iterative model reduction, contributing to the low number of feasible solutions found.

Table 3. Computational results aggregated by instance category

	CP without iterative reduction				CP with iterative reduction			
Category	Feas. sol.	Opt. sol.	Opt. gap avg	t (s) avg	Feas. sol.	Opt. sol.	Opt. gap avg	t (s) avg
smi_close	9/9	7/9	13%	144.2	9/9	9/9	0%	17.9
smi_headway	12/12	6/12	21%	336.1	12/12	10/12	12%	123.5
nor1_critical	10/10	10/10	0%	0.3	10/10	10/10	0%	0.1
nor1_full	10/10	9/10	3%	229.8	10/10	10/10	0%	128.0
nor2	5/5	5/5	0%	8.7	5/5	5/5	0%	5.6
nor3	5/5	5/5	0%	13.0	5/5	5/5	0%	5.4
nor4_large	0/5	0/5	100%	600.0	2/5	0/5	97%	600.0
nor4_small	0/5	0/5	100%	600.0	5/5	0/5	75%	600.0
nor5_large	0/5	0/5	100%	600.0	0/5	0/5	100%	600.0
nor5_small	0/5	0/5	100%	600.0	5/5	0/5	99%	600.0
swi	4/9	3/9	67%	440.1	5/9	5/9	44%	381.8
wab_large	0/16	0/16	100%	600.0	16/16	0/16	100%	600.0
wab_small	0/16	0/16	100%	600.0	16/16	0/16	91%	600.0
Total	55/112	45/112	55%	383.1	100/112	54/112	49%	335.8

The presented solution method with iterative model reduction achieved competitive results in the DISPLIB 2025 competition [14]. It obtained the fourth-highest overall score among the 15 participating teams and achieved 58 best-known solutions. These results further support the practical effectiveness of the proposed model reduction method.

We also evaluate the effect of the iterative model reduction method directly on model size in Table 4. On average over all instances, the preprocessing reduces the number of operations by 45%, potential conflicts by 53%, CP variables by 50%, and CP constraints by 22%, while requiring only an average of 2.7 s of additional preprocessing time. This confirms that the proposed reductions substantially simplify the resulting CP model at a relatively low computational cost. The largest reduction is observed in the number of potential conflicts, reaching

84% on average in one instance category. In contrast, the relatively smaller reduction in the number of CP constraints is explained by the fact that the solution method with iterative reduction can use the auxiliary deadlock-prevention constraints, whereas these cannot be applied without operation graph linearization.

Table 4. Effect of iterative reduction on model size

Category	Without iterative reduction				With iterative reduction				
	Op. avg	Conf. avg	CP var. avg	CP constr. avg	Op. reduct. avg	Conf. reduct. avg	CP var. reduct. avg	CP constr. reduct. avg	Extra prep. t (s) avg
smi_close	1318	13069	28368	63395	74%	84%	83%	69%	0.0
smi_headway	1413	13092	28559	63493	70%	81%	67%	54%	0.0
nor1_critical	464	1169	3089	6772	38%	41%	37%	14%	0.0
nor1_full	6304	257835	526005	1176102	38%	45%	44%	7%	0.9
nor2	1586	10447	23498	53596	43%	45%	43%	12%	0.2
nor3	1290	9636	21308	48504	35%	37%	36%	5%	0.1
nor4_large	27825	2336507	4720977	10532750	49%	61%	60%	20%	13.0
nor4_small	17238	765103	1559799	3481042	49%	61%	60%	20%	3.8
nor5_large	43416	3949489	7974124	17671460	50%	67%	66%	25%	26.1
nor5_small	26945	1500954	3048549	6770802	50%	66%	66%	25%	7.7
swi	20333	13328455	26679682	54533588	34%	58%	57%	27%	2.9
wab_large	5802	126197	261547	593399	36%	37%	36%	7%	0.4
wab_small	3422	41976	89315	204905	35%	35%	33%	4%	0.1
Total	9094	1503324	3021082	6335064	45%	53%	50%	22%	2.7

6 Conclusions and Future Work

In this paper, we presented an iterative model reduction method for the train dispatching problem, together with a constraint programming formulation based on the reformulated model. We introduced a series of model transformations that tighten lower bounds, merge operations, prune dominated subpaths, remove redundant resources, and linearize the operation graphs while preserving optimal solutions. We applied the iterative model reduction as a preprocessing step and formulated a compact CP model based on the resulting linearized operation graphs. Computational results on the DISPLIB 2025 benchmark library show that the proposed model reduction method can substantially simplify large train dispatching instances and improve performance in practice. The presented solution method was able to find high-quality feasible solutions for most benchmark instances.

As part of future work, we aim to introduce additional pruning transformations to further reduce the size of problem instances. We also plan to develop a branch-and-bound solution method built on the proposed iterative model reduction, where reductions are applied dynamically after each branching decision in order to narrow the search space.

Acknowledgements The project supported by the Doctoral Excellence Fellowship Programme (DCEP) is funded by the National Research Development and Innovation Fund of the Ministry of Culture and Innovation and the Budapest University of Technology and Economics.

References

1. Cordeau, J.-F., Toth, P., Vigo, D.: A survey of optimization models for train routing and scheduling. Transportation Science **32**(4), 380–404 (1998)
2. D'Ariano, A., Pacciarelli, D., Pranzo, M.: A branch and bound algorithm for scheduling trains in a railway network. European Journal of Operational Research **183**(2), 643–657 (2007)
3. D'Ariano, A., Pranzo, M., Hansen, I.A.: Conflict resolution and train speed coordination for solving real-time timetable perturbations. IEEE Transactions on Intelligent Transportation Systems **8**(2), 208–222 (2007)
4. Mannino, C., Mascis, A.: Optimal real-time traffic control in metro stations. Operations Research **57**(4), 1026–1039 (2009)
5. Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M.: Centralized versus distributed systems to reschedule trains in two dispatching areas. Public Transport **2**(3), 219–247 (2010)
6. Agasucci, V., Grani, G., Lamorgese, L.: Solving the train dispatching problem via deep reinforcement learning. Journal of Rail Transport Planning & Management **26**, 100394 (2023)
7. Schälicke, M., Nachtigall, K.: Solving the real-time train dispatching problem by column generation. Transportation Science **59**(3), 587–602 (2025)
8. Schutt, A., Cardellini, M., Dekker, J.J., Harabor, D., Maratea, M., Vallati, M.: Constraint-Based In-Station Train Dispatching. In: Proceedings of the 31st International Conference on Principles and Practice of Constraint Programming (CP 2025), LIPIcs, vol. 340, pp. 33:1–33:24 (2025)
9. Song, Y., Chen, B., Ye, W., Wang, T.: An efficient hybrid method for dispatching trains at complex railway stations. Computers & Industrial Engineering **210**, 111512 (2025)
10. Zwaneveld, P.J., Kroon, L.G., van Hoesel, S.P.M.: Routing trains through a railway station based on a node packing model. European Journal of Operational Research **128**(1), 14–33 (2001).
11. Vansteenwegen, P., Dewilde, T., Burggraeve, S., Cattrysse, D.: An iterative approach for reducing the impact of infrastructure maintenance on the performance of railway systems. European Journal of Operational Research **252**(1), 39–53 (2016).
12. Lamorgese, L., Mannino, C.: An exact decomposition approach for the real-time train dispatching problem. Operations Research **63**(1), 48–64 (2015).
13. Leutwiler, F., Corman, F.: A review of principles and methods to decompose large-scale railway scheduling problems. EURO Journal on Transportation and Logistics **12**, 100107 (2023).
14. Kloster, O., Luteberget, B., Mannino, C., Sartor, G.: DISPLIB: a library of train dispatching problems. arXiv preprint arXiv:2509.12254 (2025). https://arxiv.org/abs/2509.12254
15. Perron, L., Didier, F.: CP-SAT. Version 9.12. Google (2025). https://developers.google.com/optimization/cp/cp_solver/

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Extending the Minimal Perturbation Problem with Disruptiveness Weighting for Dynamic University Timetabling

Helen Kirby-Hawkins

Manchester Metropolitan University, Manchester, UK
hkirbyhawkins@gmail.com

Abstract. University course timetabling is a dynamic optimization problem in which published schedules frequently require modification due to changes in enrolment, staffing, or resource availability. The Minimal Perturbation Problem (MPP) addresses this by seeking revised timetables that minimize deviation from an existing solution. While many standard formulations of MPP incorporate weighted measures of perturbation, these approaches do not typically account for how disruption varies with the timing of a change within the teaching period. This paper proposes a behaviorally informed extension to MPP that introduces a disruptiveness weighting for post-publication timetable changes. The proposed Minimally Disruptive Perturbation (MDP) model weights post-publication changes according to the week of occurrence, the type of change, and the number of students affected. This enables a more nuanced assessment of disruption in dynamic timetabling contexts. The model incorporates temporal weighting, change type and student reach to assess the behavioral cost, through a Post-Publication Impact Score (PPIS), of timetable volatility. Using institutional data from a large UK university, the paper demonstrates how the PPIS differentiates between changes that are operationally similar but behaviorally distinct. The approach is intended as a decision-support metric that can be incorporated into post-publication rescheduling processes and candidate solution evaluation, supporting more human-centric timetable optimization.

Keywords: *University timetabling, minimal perturbation problem, dynamic scheduling, disruptiveness weighting, human-centric optimization.*

1 Introduction

University course timetabling is commonly formulated as a constraint-based optimization problem in which limited resources—rooms, staff, and time slots—must be allocated while satisfying hard and soft constraints. While much of the literature focuses on the generation of feasible initial timetables, in practice university timetables remain subject to frequent change after publication. Fluctuations in student enrolment, late staffing changes, and external constraints often render published timetables infeasible, requiring re-optimization post-publication throughout the teaching semester.

The Minimal Perturbation Problem (MPP) was introduced to address this challenge by prioritizing solutions that minimize deviation from an existing timetable when changes are required. MPP has been widely adopted in dynamic timetabling environments and is typically operationalized by minimizing the number of changes or constraint violations relative to the original solution. While this approach is effective in preserving structural similarity, it assumes that all changes are equally disruptive, thereby overlooking their behavioral impact. In operational university settings, this assumption is problematic. A room change late in the semester is treated equivalently to a day or time change in the first week of teaching, despite the latter often having greater consequences for students and staff. Timetabling practitioners frequently recognize this distinction informally, but it is rarely reflected in optimization models or decision-support tools. More fundamentally, this approach implicitly privileges the existing timetable by penalizing deviation from it, effectively treating the published solution as a desirable reference state, even where it may not represent an optimal or stable schedule.

This paper proposes an extension to the MPP framework that incorporates disruptiveness weighting into post-publication timetable changes. The proposed Minimally Disruptive Perturbation (MDP) model introduces a quantitative metric that weighs changes according to their timing, type, and scale of impact. The contribution of this paper is not the introduction of weighting per se, but the explicit integration of temporal sensitivity into the evaluation of post-publication timetable changes, demonstrating how disruption varies with timing, type, and scale.

2 Related Work

2.1 University Timetabling and Scheduling

University course timetabling is a well-established problem in operations research, often framed as a constraint-based optimization challenge. The goal is to allocate limited resources — such as rooms, instructors, and time slots — while satisfying a set of hard and soft constraints (Lindahl et al., 2018).

Automated scheduling systems have become increasingly sophisticated, employing techniques such as integer programming (Phillips et al., 2017), genetic algorithms (Abdelhalim and El Khayat, 2016), and satisfiability solvers (Lemos et al., 2020) to generate feasible timetables.

Many real-life timetabling problems can be codified as constraint-based optimization problems. Tolerating dynamism in the university timetable is systematically managed by algorithms solving optimization problems, which are subject to disruptions that make the original solution no longer feasible, so the problem becomes dynamic and requires re-solving with minimal perturbations (Lemos et al., 2020).

However, these models typically prioritize computational efficiency and feasibility over human experience. The Minimal Perturbation Problem (MPP), introduced by Müller et al. (2005), addresses the challenge of modifying an existing timetable with minimal disruption. While MPP has been widely adopted in dynamic scheduling environments, it primarily focuses on minimizing the number (count) of changes or

constraint violations, without considering their disruptive weight and effect on objects in the timetable (staff, students and resources). There is an anecdotal belief that not all types of changes or constraints are equally weighted. Many models account for hard and soft constraints, but these are not sufficiently nuanced to reflect weighting or the human behavioral impact of changes to the timetable when evaluating the best fit of a candidate option as a potential solution to the timetabling problem. Unlike soft constraint weighting within an optimization model, the proposed PPIS operates as an evaluative metric applied to candidate post-publication solutions, rather than directly influencing solver search.

However, empirical and computational studies demonstrate that strict minimization of perturbations can lead to low quality solutions, and that allowing a modest increase in change often yields substantially improved outcomes (Phillips et al., 2017). Moreover, MPP formulations require institution-specific weighting policies that are external to the model itself (Kingston, 2016), and many approaches struggle to efficiently accommodate common real-world disruptions such as domain or capacity changes (Banbara et al., 2019). Collectively, this body of work suggests that classical MPP formulations prioritize structural similarity over the qualitative and behavioral impact of timetable changes on end users, motivating the need for richer, human-centered disruption models in university contexts. There also remains a gap in the literature regarding how timetabling changes may causally affect student behavior, particularly in the early weeks of a semester when new routines are still forming.

Recent work has highlighted the importance of human-centric consideration in timetabling, recognizing that schedules are consumed by real students — human end-users with complex lives — rather than abstract "student" objects, of which the literature speaks in somewhat reductionistic terms. Lindahl et al. (2018) advocate for a strategic view of timetabling that includes institutional goals and user preferences.

It is worth situating this contribution precisely against existing weighted-perturbation approaches, since weighting change severity is not itself new: El Sakkout et al. (1998) introduced distance-based measures of perturbation in dynamic scheduling, Müller et al. (2005) and the resulting UniTime system support a weighted sum of criteria including the number of affected students and the type of change, and several institution-specific MPP deployments already apply some form of change-type weighting in practice (Kingston, 2016). What these existing weighting schemes do not incorporate is an explicit temporal term: a given change type and student count are weighted identically regardless of whether the change occurs in week one or week eight of teaching. The contribution of this paper is therefore not the use of weighting per se, but the introduction of a temporal-sensitivity term (W) that decays with elapsed teaching time, combined multiplicatively with type and reach so that early, compound, high-reach changes are penalized disproportionately rather than additively. This paper further grounds the type and reach components empirically against institutional change data and student-reported disruption rankings from a single UK university, rather than relying solely on practitioner judgement, although the specific weight values themselves remain provisional (Sect. 4.3). Existing weighted approaches therefore capture structural differences between solutions, but do not explicitly model how the context in which a change is experienced alters its practical impact. These approaches therefore measure perturbation as a property of the solution

structure, rather than as a function of when and how changes are experienced by end-users.

This study contributes to this discourse by extending the MPP framework and principles to consider behavioral dimensions, offering a Post-Publication Impact Score (PPIS) that quantifies the impact of changes based on timing, type, and student reach. This formal weighting mechanism is reflective of the practical distinctions made between types of changes made by practitioners and offers a supportive model for decision-making.

3 Problem Context and Data

The study is situated within a large UK university operating a centrally managed timetable and was conducted between December 2024 and March 2025. Post-publication timetable changes occur frequently due to fluctuations in student numbers, module enrolment changes, staff amendments and room changes. Institutional timetable change data were extracted for a single academic semester, recording the week of change, the type of change, and the number of students affected. The analysis focuses on changes occurring within the first eight teaching weeks, where timetable volatility is highest. The dataset included all students on activities affected by two or more timetable changes — excluding minor administrative edits such as a change of named staff member or a cosmetic relabeling of an activity's display name, neither of which affects when, where, or how a student attends — in the first four weeks of teaching (N = 10,341). Student instances in this dataset do not correspond to unique students, as individual students may be affected by multiple timetable changes.

3.1 Quantitative Findings

Analysis of the institutional data reveals that 49.4% of the entire student population experienced at least one impactful timetable change within the first eight weeks of teaching. This equates to 21,991 unique students, with 11,852 experiencing two or more changes. A total of 906 changes and 747 additions or deletions were recorded during the period, affecting 46,940 student instances (acknowledging that unique students in this count will have been affected more than once, hence the total exceeding the population).

During the first four weeks of teaching, 10,341 students were impacted by timetable changes. These early disruptions are particularly significant given the importance of this period for habit formation.

Activity and Course-Level Disruption Analysis. To further analyze the nature of timetable volatility, an analysis was conducted on the types of activities and which courses were most frequently affected by timetable changes, drawn from the same institutional dataset. Using the activity naming conventions available in the dataset, activity types were examined from the "activity name" field. The most frequently changed activity types were:

Table 1. Total changes by activity type, number of changes in weeks 1–4, and percentage of total changes occurring in weeks 1–4, by activity type.

Activity Type	Total Changes	Changes in Weeks 1–4	% in Weeks 1–4
Seminar	205	153	74.63%
Lecture	192	124	64.58%
Tutorial	114	88	77.19%
PC session	111	45	40.54%
Workshop	91	70	76.92%
Practical	36	23	63.89%
Drop-In	28	22	78.57%
Web	20	9	45.00%
Groupwork	15	13	86.67%
Peer Assisted Learning (PAL)	13	13	100.00%

The table illustrates that the most frequently changed activity types are Seminars and Lectures. However, in weeks 1–4 the activity types with the highest percentage of early-week changes are PAL sessions and Groupwork. In Fig. 1 we can see the frequency of different types of change, including compound changes (such as day + time + room together), across the teaching weeks.

It should be noted that Table 1 reports raw change counts rather than change rates, and so the prominence of Seminars and Lectures partly reflects the fact that these are the most common activity types in the timetable overall, rather than indicating that these activity types are disproportionately prone to change. A more informative comparison would express changes as a proportion of total scheduled instances per activity type (i.e., a change rate), which would allow activity types to be compared on a like-for-like basis; this denominator was not available at the time of writing and is noted as a limitation of the current analysis.

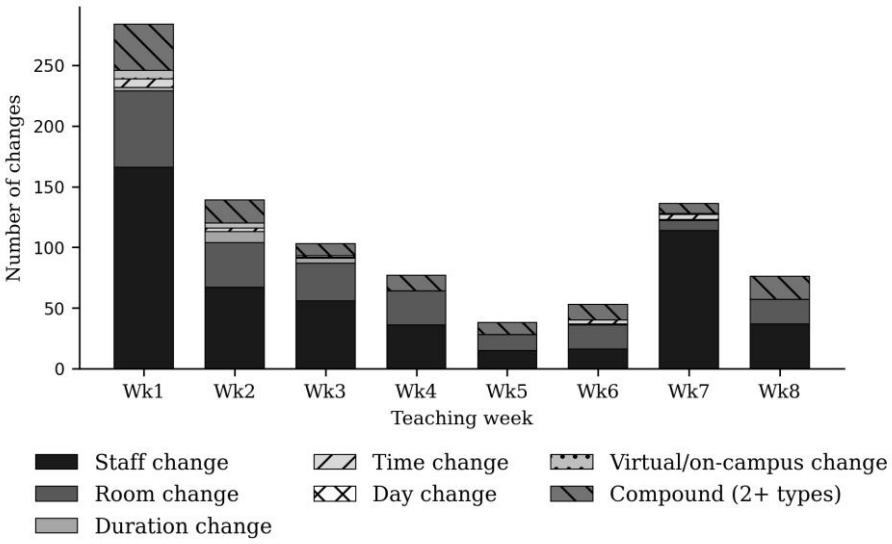


Fig. 1. Weekly distribution of timetable changes by change type, teaching weeks 1–8. Staff and room changes dominate volume throughout, while compound changes (combining two or more change types) and day/time changes, though far less frequent, are concentrated in the earliest teaching weeks.

A secondary pattern visible in Fig. 1 is the spike in staff changes in Week 7, which is comparable in scale to the Week 1 peak; this coincides with a point in the semester where staff cover arrangements are commonly renegotiated and is noted here as a candidate explanation rather than a confirmed cause. These patterns reinforce the importance of temporal context, since a substantial proportion of disruption occurs during the earliest teaching weeks, where changes are likely to have a disproportionate practical impact.

4 Disruptiveness-Weighted Minimal Perturbation

4.1 Limitations of Standard MPP

The standard MPP formulation minimizes the number of changes required to restore feasibility, treating changes primarily through structural measures of deviation, without explicit consideration of their substantially different behavioral impact. Institutional data and feedback from students indicate that this assumption obscures significant differences in practical impact relating to timing, change type and scale.

One source of this differential impact is that many students hold commitments external to their studies — part-time employment, caring responsibilities, or commuting arrangements — that are scheduled around the published day and time of an activity. A change to the day or start time of a session therefore risks conflicting with a commitment the student cannot easily move, whereas a change of room alone leaves the

student's wider schedule intact and typically requires only that they walk to a different location. This asymmetry is reflected in the type weighting T, where day and time changes are assigned higher base weights than room changes alone (Sect. 4.3).

4.2 The Post-Publication Impact Scoring Method

A Post-Publication Impact Score (PPIS) is defined for each timetable change as the product of week weighting, change type weighting, number of students affected and number of weeks affected. Earlier weeks and more compound change types are assigned higher weights. By weighting changes more heavily in earlier weeks and assigning higher scores to more disruptive change types (e.g., day and time changes), the model captures both the temporal and behavioral dimensions of disruption. This approach extends the Minimal Perturbation Problem (MPP) in the timetabling literature by incorporating student-centered metrics, offering a more holistic view of what constitutes a "minimally disruptive" change. This formulation can only be meaningfully applied to post-publication changes, where a student was aware of the previous timetable, since disruption is defined relative to a prior, known state.

A clarification is required regarding the temporal reference point used to determine W. In this model, the week weighting is fixed at the point a change is issued to students: it reflects how early in the teaching period a given disruption lands, independent of how many prior changes that same activity may already have undergone. Where an activity is changed more than once, each change is scored independently at its own issue-week, rather than being measured cumulatively against the originally published timetable. This choice keeps the model usable in an operational setting, since schedules must be updated promptly and a single, fixed point of comparison — the original publication — would assign identical, undiminishing penalties to every subsequent change regardless of when in the term it occurred. The trade-off is that the current formulation does not yet capture the cumulative burden on a student who experiences several smaller changes to the same activity across consecutive weeks; this longitudinal dimension is identified as a direction for future work in Sect. 5. This ensures that temporal sensitivity reflects when disruption is experienced by students, rather than distance from a fixed baseline timetable. This distinction implies that candidate timetable repairs that are equivalent under perturbation-count metrics may be ranked differently when evaluated using a disruptiveness-weighted approach.

4.3 Post-Publication Impact Score (PPIS) Formula

The Post-Publication Impact Score (PPIS) is defined as follows, expressed as the product of four component weights:

$$PPIS = W \times T \times S \times N \tag{1}$$

that is, post-publication impact equals the week weighting multiplied by the type-of-change weighting, the number of students affected, and the number of weeks the change persists, where:

W — the week weighting, decreasing as the teaching semester progresses (W = 8 if the change is issued in Week 1, falling by 1 for each subsequent week, down to W = 1 in Week 8);

T — the type-of-change weighting, derived from the base weights of the individual change types involved (see below);

S — the number of students affected by the change;

N — the number of weeks the change remains in effect.

The total post-publication impact of a proposed set of changes δ is then the sum of the individual PPIS values:

$$min\delta \ \Sigma PPISi \ = \ \Sigma (Wi \times Ti \times Si \times Ni) \tag{2}$$

where δ is a proposed set of timetable changes, PPISi is the Post-Publication Impact Score of change i, and the summation runs over all changes i = 1, …, n in the set.

This formula captures temporal sensitivity through the week of the change, qualitative severity through the type weighting, scale of impact through the number of students affected, and the duration of impact through the number of weeks affected. For compound changes, the type weighting T is computed as the product of the individual change-type weights and is therefore multiplicative rather than additive.

The four component weights (W, T, S and N) are combined multiplicatively rather than additively so that a change which is severe on every dimension — early, compound, and affecting a large cohort — is scored disproportionately higher than a change that is merely severe on one dimension, consistent with the practitioner intuition that such changes are qualitatively, not just additively, worse. The base type weights underlying T in the current model (Day change = 4.97, Time change = 4.46, Duration change = 4.46, Room change = 3.07, Staff change = 2.13, Virtual/on-campus change = 2.07) were assigned by the author based on practitioner judgement and informal student feedback gathered during the study, rather than derived through formal statistical calibration against survey data; they are presented as an illustrative starting scheme rather than a validated weighting, and institutions adopting the model would be expected to calibrate these values against their own student feedback. A consequence of the multiplicative structure is that no individual weight may be set to zero, since this would zero the PPIS regardless of the severity of the other components; all weights in the current scheme are therefore bounded below by approximately 2.

4.4 Patterns of Post-Publication Impact Across Units

The PPIS was computed for every recorded change in the dataset. Scores were highly right-skewed: across the 50 highest-scoring changes, scores ranged from 4,516 to 77,645, with a median of 7,738 and a mean of 12,506 — indicating that a small number of changes contributed disproportionately to total post-publication impact relative to the bulk of the distribution. The single highest-scoring change in the dataset was a combined room, time and day change to a first-year tutorial in Week 2, affecting 163 students (PPIS: 77,645). By contrast, a Peer Assisted Learning session affected in the same week by only a room and time change, but reaching a substantially larger cohort

of 560 students, scored lower overall (53,673) — illustrating that compound change type can outweigh raw student reach in the PPIS.

The application of the PPIS to institutional data revealed stark differences in how disruption is distributed across the curriculum, and demonstrated that frequency of change does not equal total post-publication impact. Some units in the dataset emerged as particularly volatile, with scores exceeding 6,000 points (within the upper quartile of observed changes in the dataset) — placing them among the top quartile of recorded changes by PPIS. These units experienced multiple compound changes (combinations of day, room, and time) in the early weeks — the types of changes that students consistently ranked as most disruptive. This demonstrates that changes which are similar in frequency may differ substantially when evaluated using the PPIS.

Interestingly, units with high student enrolments were not always the most disruptive. This suggests that the timing and type of change may be more critical than scale alone: a single day change in Week 1, affecting a modest cohort, may be more behaviorally disruptive than a room change in Week 8 for a lecture affecting hundreds of students. This insight challenges traditional assumptions in timetabling optimization, which often prioritize minimizing the number of affected students without considering the wider behavioral and contextual factors of the change — a gap the proposed model is intended to address.

4.5 Implications for Timetabling Practice and Policy

The findings of this study have clear implications for institutional policy and practice. Primarily, universities should prioritize timetable stability in the first four weeks of teaching to reduce disruption. This could involve introducing a change freeze for timetables during this period, improving forecasting for student enrolments, and adopting better staff workload-planning models. Where changes are unavoidable, institutions should aim to minimize behavioral impact by avoiding highly disruptive change types (especially compound day-and-time changes), providing adequate notice, and offering online alternatives where reasonably practical. These implications arise directly from the observed concentration of high-disruptiveness changes in early teaching weeks and from the asymmetry between change types.

5 Conclusion

This paper extends the Minimal Perturbation Problem by introducing a disruptiveness-weighted model for post-publication university timetabling. Empirical analysis demonstrates that changes which are operationally similar can differ substantially in behavioral impact. By quantifying this impact, the MDP model provides a practical decision-support metric that complements existing optimization approaches and supports more human-centric scheduling decisions.

In a heuristic-based timetabling system, the MDP model would be most effectively deployed within the post-publication scheduling window, during the candidate evalu-

ation and selection phase of the heuristic loop — specifically when deciding which change to make among several feasible options.

Moreover, the scoring model could be extended to include additional variables, such as notice period provided, activity type, and cumulative exposure to changes (staff or student). For example, a student who experiences minor changes in three different weeks may incur more cumulative disruption than a student who experiences a single major change. The results suggest that incorporating temporal sensitivity into perturbation evaluation offers a practical means of distinguishing between structurally minimal and behaviorally minimal disruption in dynamic university timetabling. In doing so, the model highlights a limitation of perturbation-based approaches: that proximity to an existing solution is not necessarily a reliable proxy for minimal disruption in practice.

References

1. Abdelhalim, E.A., El Khayat, G.A.: A utilization-based genetic algorithm for solving the university timetabling problem (UGA). Alexandria Engineering Journal 55(2), 1395–1409 (2016).
2. Agarwal, A., McGuire, J., Webb, J.: Understanding the impact of timetable scheduling on university students' academic performance. In: 2023 STARS Conference, Brisbane (2023).
3. Banbara, M., Inoue, K., Kaufmann, B., Okimoto, T., Schaub, T., Soh, T., Tamura, N., Wanko, P.: teaspoon: solving the curriculum-based course timetabling problem with answer set programming. Annals of Operations Research 275(1), 3–37 (2019).
4. El Sakkout, H., Richards, T., Wallace, M.: Minimal perturbation in dynamic scheduling. In: Proceedings of the 13th European Conference on Artificial Intelligence (ECAI 1998), pp. 504–508. John Wiley & Sons, Brighton, UK (1998).
5. Kingston, J.H.: Specifying and solving minimal perturbation problems in timetabling. In: Proceedings of the 11th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2016), pp. 1–14 (2016).
6. Lemos, A., Monteiro, P.T., Lynce, I.: Minimal perturbation in university timetabling with maximum satisfiability. In: International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research, pp. 317–333. Springer, Cham (2020).
7. Lindahl, M., Mason, A.J., Stidsen, T., Sørensen, M.: A strategic view of university timetabling. European Journal of Operational Research 266(1), 35–45 (2018).
8. Müller, T., Rudová, H., Barták, R.: Minimal perturbation problem in course timetabling. In: Burke, E.K., Trick, M.A. (eds.) Practice and Theory of Automated Timetabling V. LNCS, vol. 3616, pp. 126–146. Springer, Heidelberg (2005).
9. Phillips, A.E., Walker, C.G., Ehrgott, M., Ryan, D.M.: Integer programming for minimal perturbation problems in university course timetabling. Annals of Operations Research 252(2), 283–304 (2017).

Personnel Rostering in a Public Library: A Case Study

Bruno Basckeira Chinaglia[1], Douglas Nogueira Nascimento[1 [0000−0001−9278−505X]], Greet Vanden Berghe[2 [0000−0002−0275−5568]], and Franklina Maria Bragion Toledo[1 [0000−0003−2823−7600]]

[1] Institute of Mathematical and Computer Sciences, ICMC-USP, Brazil
[2] Department of Computer Science, KU Leuven, Belgium
brunobchinaglia@usp.br, douglasnn92@gmail.com,
greet.vanden.berghe@kuleuven.be, fran@icmc.usp.br

Abstract. This paper addresses a real-world personnel rostering problem arising in a public university library. The roster covers Monday–Saturday operations and must satisfy shift-coverage and skill-mix requirements, while handling a set of undesirable duties (Friday evenings and Saturdays) subject to rotation and adjacency rules. The main challenge is to reconcile these operational constraints with staff preferences regarding undesirable duties and to do so equitably. We introduce a max–min fairness criterion that maximises the minimum normalised *happiness* across employees, thereby protecting the least satisfied staff member with respect to the roster accommodating their preferences. We first investigate the effectiveness of a compact mixed-integer linear programming (MILP) formulation for this setting. We also propose an alternative, structure-exploiting solution workflow based on a two-stage decomposition: Stage 1 optimises assignments on the undesirable days, and Stage 2 completes weekday evening duties using the fixed weekly availability patterns implied by the Stage 1 decisions. Since weeks with holidays may induce additional imbalance in days off, we further introduce a refinement model that balances worked days within holiday weeks by minimising the range between the most- and least-loaded staff members. A real-world application shows that the proposed workflow produces equitable and transparent rosters, with high worst-case normalised satisfaction, and can be used routinely by non-technical personnel.

Keywords: Personnel rostering · Mixed-integer linear programming · Decomposition approach

1 Introduction

Personnel rostering problems are widespread in service operations and have been extensively studied in contexts such as hospitals [2, 12], schools and universities [3], and many other sectors [5, 8]. In this paper, we address a real rostering problem at the ICMC/USP Library. The responsible librarian used to prepare this roster manually, which was a very time-consuming task. Despite significant

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

efforts, the team did not always perceive the roster as fair. The problem requires reconciling coverage and skill-mix requirements with staff availability and preferences under a variety of institutional rules. Such decisions directly affect service quality, the fair allocation of undesirable duties (weekend and evening assignments), and staff satisfaction [6].

Generating good-quality solutions for personnel rostering can benefit from advances in modelling and solution methods [5]. Survey and tutorial contributions highlight that practical approaches typically rely on carefully designed models, often within a mathematical programming framework that deals with feasibility constraints [7] and objective components, for example soft constraint violations [9]. They also emphasise that exploiting problem structure, focusing on a limited set of undesirable duties, or separating independent decisions is a key driver of scalable, implementable solution workflows [11]. Beyond feasibility and solution quality, practical adoption also depends on transparency and perceived fairness. For deployment in practice, decision makers often require solutions that are understandable and whose allocations can be justified to staff, especially when considering personal preferences [1].

We consider a personnel rostering problem where staff members, based on their skills, belong to either the Information Processing (IP) or the User Services (US) group. Apart from this division, the staff members in the ICMC/USP library also belong to one of three categories: *support staff* (4), *technicians* (4) and *librarians* (3), based on their level of education and training. Support staff handle routine tasks such as public service, and their schedules are fixed. These staff members can therefore be omitted from the rostering problem. The remaining staff must be assigned to weekday evening and Saturday shifts over a semester-long horizon.

Besides standard coverage and skill-mix requirements, the roster must maximise a fairness objective which depends on the distribution of Friday evening and Saturday assignments, which are considered undesirable. Interestingly, staff members explicitly state preferences for these undesirable shifts, and the library requires these duties to be allocated transparently and equitably. It is important to enforce a rotation of these undesirable shifts when modelling the problem. In addition, the occurrence of consecutive Friday–Saturday assignments should be restricted. Compared with personnel rostering models that primarily minimise soft constraint violations or maximise total preference satisfaction, we explicitly optimise equity through a max–min objective that maximises the minimum normalised satisfaction across employees, thereby protecting the least satisfied staff member and producing allocations that are easy to justify to practitioners.

While personnel rostering problems are often assumed to be NP-hard, recent theoretical work shows that tractability can depend strongly on the exact constraint set and modelling granularity, and that several meaningful rostering variants admit polynomial-time formulations via network flow models [10]. Motivated by these insights, we investigate which components drive difficulty in the ICMC/USP Library setting, with particular emphasis on (i) whether a compact mixed-integer linear programming (MILP) approach can solve the problem di-

rectly, and how the inclusion of an explicit max–min fairness objective affects solution structure and practical solvability; and (ii) whether exploiting the separation between undesirable duties (Friday evenings and Saturdays) and weekday completion enables a modular, implementable workflow. This perspective also motivates our decomposition and refinement steps, which isolate undesirable-day decisions and mitigate additional imbalances induced by holidays.

The remainder of this paper is organised as follows. Section 2 formalises the problem and its operational rules. Section 3 presents a compact mixed-integer linear programming formulation, while Section 4 describes an alternative two-stage decomposition approach. Section 5 introduces an improvement strategy to handle holidays. Section 6 reports the computational results and evidence from deployment. Finally, Section 7 concludes the paper and points to future work.

2 Problem description

The personnel rostering problem involves assigning employees to shifts over a finite planning horizon to meet demand coverage, while adhering to institutional and contractual rules (hard constraints). Due to the possibility of omitting regular day shifts from the problem, coverage and assignments can be expressed per day. The planning horizon follows a Monday–Saturday schedule. The problem focuses on the assignment of undesirable days (i.e., Friday evenings and Saturdays). Employee preferences for these days are treated as soft constraints and balanced using fairness-based objectives. Although simple cyclical rotations are attractive because of their transparency and low implementation effort, they would be restrictive for the librarian case study. Friday–Saturday adjacency restrictions, holiday effects, and day-specific employee preferences cannot be met with a single repeating rotation without compromising feasibility or fairness.

The rostering decisions concern assigning technical and librarian staff to weekday evening and Saturday shifts, while respecting the following requirements: (i) every weekday and Saturday must be covered by two staff members at technician or librarian level, (ii) Saturdays must be covered by one staff member from each group (IP and US), (iii) assignments on Fridays and Saturdays must follow a rotation. Throughout the paper, we use the term *rotation* to denote a policy that distributes the undesirable assignments among eligible staff over the planning horizon.

After a series of meetings, the staff members involved in the rostering process requested the incorporation of additional *hard constraints* for the Friday and Saturday rotations to avoid repeatedly assigning the same employee to undesirable days in consecutive weeks. In particular, (iv) no employee may be assigned to three consecutive Fridays, (v) no employee may be assigned to three consecutive Saturdays, and (vi) an employee assigned to a Friday must not be assigned to the immediately following Saturday.

Staff also stated their fixed *availability pattern* for weekdays to be followed throughout the planning horizon. Three patterns are allowed: (1) Monday, Tuesday and Friday; (2) Tuesday, Wednesday and Friday; and (3) Wednesday, Thurs-

day and Friday. Each employee is associated with exactly one of these patterns for the entire rostering horizon. In any given week, each employee may be scheduled for at most two of the three days specified by their availability pattern. The librarian scheduling problem belongs to the category $SabB|2|L$ according to [4].

3 Mathematical Model

This section presents a compact MILP formulation for the library personnel rostering problem described in Section 2. Our formulation captures the assignment of technicians and librarians over the planning horizon, while enforcing the constraints described in the previous section. In addition, it incorporates individual preference scores for working on undesirable days and pursues a max–min fairness criterion, i.e., it maximises the preference satisfaction of the least satisfied staff member. Staff members provide preference ratings for each Friday evening and Saturday. These preference ratings are encoded as integer scores HF_{id} and HS_{id}, with 1 = dislike, 2 = indifferent, and 3 = like.

The proposed model, here denoted by *LRM (Library Rostering Model)*, is represented by (1)–(21). For clarity, in Table 1, we first introduce the sets, the parameters and the decision variables used throughout the formulation.

Table 1: Sets, parameters and variables for the LRM.

Sets	
ST	set of staff members ($ST = \{1, \ldots, \|ST\|\}$);
IP	set of staff members in the Information Processing department where $IP \subset ST$;
US	set of staff members in the User Services department where $US \subset ST$, $IP \cup US = ST$ and $IP \cap US = \emptyset$;
M	set of months in the planning horizon ($M = \{1, \ldots, \|M\|\}$);
D	set of days in the planning horizon ($D = \{1, ..., \|D\|\}$);
$\overline{D}$	set of non-working days in the planning horizon;
F_m (S_m)	set of Fridays (Saturdays) in month $m \in M$;
$NF(NS)$	set of Fridays (Saturdays) in the horizon ($NF = \cup_{m \in M} F_m$ and $NS = \cup_{m \in M} S_m$);
$\overline{NF}(\overline{NS})$	set of all Fridays (Saturdays) except last two of the horizon.
Parameters	
AF_m	average number of Fridays per staff member in month $m \in M$ $\left(AF_m = \frac{2\,\|F_m \setminus \overline{D}\|}{\|ST\|}\right)$, the factor 2 reflects the coverage constraint of two employees per Friday;
$A2F_m$	average number of Fridays per staff member over the consecutive months m and $m+1$, $m \in M \setminus \{\|M\|\}$ $\left(A2F_m = AF_m + AF_{m+1}\right)$;
AHF	average number of Fridays per staff member over the horizon $\left(AHF = \frac{2\,\|NF \setminus \overline{D}\|}{\|ST\|}\right)$;
$AS_m^{IP}(AS_m^{US})$	average number of Saturdays per staff member in the IP(US) group in month $m \in M$ $\left(AS_m^{IP} = \frac{\|S_m \setminus \overline{D}\|}{\|IP\|}\right)$;
$A2S_m^{IP}(A2S_m^{US})$	average number of Saturdays per staff member in the IP (US) group over the consecutive months m and $m+1$, $m \in M \setminus \{\|M\|\}$;
AHS^{IP} (AHS^{US})	average number of Saturdays per staff member in the IP (US) group over the horizon;
HF_{id} (HS_{id})	"happiness" (preference) of staff member $i \in ST$ if working on Friday $d \in NF$ (Saturday $d \in NS$);
$H_i^{\max}$	maximum attainable happiness for staff member $i \in ST$ based on their preferences.
Variables	
$f_{\max}$	continuous variable representing the minimum normalised happiness across staff;
x_{id}	equals 1 if staff member $i \in ST$ is assigned to work on day $d \in D$, and 0 otherwise.

$$\textit{Maximise} \quad f_{\max} \tag{1}$$

subject to

$$f_{\max} \leq \frac{\sum_{d \in NF \setminus \overline{D}} HF_{id}\, x_{id} + \sum_{d \in NS \setminus \overline{D}} HS_{id}\, x_{id}}{H_i^{\max}} \qquad i \in ST \qquad (2)$$

$$\sum_{i \in ST} x_{id} = 2 \qquad d \in D \setminus \overline{D} \qquad (3)$$

$$\sum_{i \in IP} x_{id} = 1 \qquad d \in NS \setminus \overline{D} \qquad (4)$$

$$\sum_{i \in US} x_{id} = 1 \qquad d \in NS \setminus \overline{D} \qquad (5)$$

$$x_{i,d-4} + x_{i,d-3} + x_{id} \leq 2 \qquad i \in ST,\ d \in NF \qquad (6)$$

$$x_{i,d-3} + x_{i,d-2} + x_{id} \leq 2 \qquad i \in ST,\ d \in NF \qquad (7)$$

$$x_{i,d-2} + x_{i,d-1} + x_{id} \leq 2 \qquad i \in ST,\ d \in NF \qquad (8)$$

$$x_{id} + x_{i,d+1} \leq 1 \qquad i \in ST,\ d \in NF \qquad (9)$$

$$x_{id} + x_{i,d+7} + x_{i,d+14} \leq 2 \qquad i \in ST,\ d \in \overline{NF} \cup \overline{NS} \qquad (10)$$

$$\lfloor AF_m \rfloor \leq \sum_{d \in F_m \setminus \overline{D}} x_{id} \leq \lceil AF_m \rceil \qquad i \in ST,\ m \in M \qquad (11)$$

$$\lfloor A2F_m \rfloor \leq \sum_{d \in (F_m \cup F_{m+1}) \setminus \overline{D}} x_{id} \leq \lceil A2F_m \rceil \qquad i \in ST,\ m \in M \setminus \{|M|\} \qquad (12)$$

$$\lfloor AHF \rfloor \leq \sum_{d \in NF \setminus \overline{D}} x_{id} \leq \lceil AHF \rceil \qquad i \in ST \qquad (13)$$

$$\left\lfloor AS_m^{IP} \right\rfloor \leq \sum_{d \in S_m \setminus \overline{D}} x_{id} \leq \left\lceil AS_m^{IP} \right\rceil \qquad i \in IP,\ m \in M \qquad (14)$$

$$\left\lfloor A2S_m^{IP} \right\rfloor \leq \sum_{d \in (S_m \cup S_{m+1}) \setminus \overline{D}} x_{id} \leq \left\lceil A2S_m^{IP} \right\rceil \qquad i \in IP,\ m \in M \setminus \{|M|\} \qquad (15)$$

$$\left\lfloor AHS^{IP} \right\rfloor \leq \sum_{d \in NS \setminus \overline{D}} x_{id} \leq \left\lceil AHS^{IP} \right\rceil \qquad i \in IP \qquad (16)$$

$$\lfloor AS_m^{US} \rfloor \leq \sum_{d \in S_m \setminus \overline{D}} x_{id} \leq \lceil AS_m^{US} \rceil \qquad i \in US,\ m \in M \qquad (17)$$

$$\left\lfloor A2S_m^{US} \right\rfloor \leq \sum_{d \in (S_m \cup S_{m+1}) \setminus \overline{D}} x_{id} \leq \left\lceil A2S_m^{US} \right\rceil \qquad i \in US,\ m \in M \setminus \{|M|\} \qquad (18)$$

$$\lfloor AHS^{US} \rfloor \leq \sum_{d \in NS \setminus \overline{D}} x_{id} \leq \lceil AHS^{US} \rceil \qquad i \in US \qquad (19)$$

$$f_{\max} \geq 0 \qquad (20)$$

$$x_{id} \in \{0, 1\} \qquad i \in ST,\ d \in D \setminus \overline{D} \qquad (21)$$

The objective function (1) of *LRM* aims to maximise the happiness level of the least satisfied staff member, as defined by Constraints (2). We distinguish Friday and Saturday preference scores (HF_{id} and HS_{id}) because these refer to different undesirable duties and are collected separately for $d \in NF$ and $d \in NS$. For each staff member $i \in ST$, the numerator of (2) represents the happiness due to undesirable duties assigned to i on Fridays and Saturdays, weighted by the corresponding preference scores. This happiness level is computed propor-

tionally, taking into account each staff member's maximum attainable happiness (parameter $H_i^{\max}$) based on their stated happiness ratings. The denominator $H_i^{\max}$ represents an upper bound on the happiness that employee i can attain under ideal assignments. This value is computed by assuming that employee i works the minimum required number of Friday and Saturday duties. Among these required duties, the model assumes employee i is assigned to the $\lfloor AHF \rfloor$ (which is equal to $\lfloor AHS^{IP} \rfloor$ or $\lfloor AHS^{US} \rfloor$) days with the highest preference scores, sorted in non-increasing order. Finally, each unassigned undesirable day receives a score of 4. Consequently, for staff members in the IP group, days off on Fridays and Saturdays contribute $4 * \left(|NF \setminus \overline{D}| - \lfloor AHF \rfloor + |NS \setminus \overline{D}| - \lfloor AHS^{IP} \rfloor \right)$ to $H_i^{\max}$, and analogously for the US group by replacing AHS^{IP} with AHS^{US}.

Constraints (3) ensure that exactly two staff members are assigned to each shift, while constraints (4)–(5) guarantee that one staff member from each group (IP and US) is assigned to work on Saturdays. Constraints (6)–(8) guarantee the weekday evening-shift availability rule. Specifically, for each Friday $d \in NF$, they limit the number of evening duties assigned within the corresponding three-day availability pattern: Monday–Tuesday–Friday (6), Tuesday–Wednesday–Friday (7), and Wednesday–Thursday–Friday (8). Constraints (9) prevent a staff member from being assigned to consecutive Friday and Saturday duties. Constraints (10) prevent three consecutive Friday/Saturday duties for any employee.

The balancing constraints (11)–(13) control the Friday rotation by ensuring that each staff member works between a minimum and maximum number of Friday evenings per month (Constraint (11)), over any two consecutive months (Constraint (12)), and over the entire planning horizon (Constraint (13)). Analogously, Constraints (14)–(19) balance Saturday duties across the IP and US groups. The constraints are stated separately because the groups have different sizes and each group must have a staff member working on Saturdays. In these constraints, expressions such as $F_m \setminus \overline{D}$ (and, analogously, $(F_m \cup F_{m+1}) \setminus \overline{D}$, $S_m \setminus \overline{D}$, and $NS \setminus \overline{D}$) denote the corresponding sets of *working* Fridays/Saturdays, obtained by excluding non-working days in $\overline{D}$. That is, the summations count only assignments on days when the library is open. Finally, Constraints (20)–(21) define the variables' domains.

LRM provides a detailed and comprehensive formulation that simultaneously incorporates operational requirements, individual preferences, and equity criteria in the allocation of duties. However, this richness of representation comes at the cost of a large number of variables and constraints in the formulation. In the real-world setting considered, the model is feasible because the staffing level and the library rules are mutually consistent: there are seven eligible staff members, each of whom may work at most two duties per week, yielding a weekly capacity of 14 assignments, whereas the library requires only 12 assignments per week. In addition, the staff is distributed across the fixed availability patterns and the two functional groups in a way that satisfies the weekday and Saturday coverage requirements.

A critical review of the model indicates opportunities to investigate alternative approaches that preserve the equity and preference-satisfaction criteria while

simplifying the formulation. These considerations motivated the development of a new solution approach described in Section 4.

4 Decomposition Approach

Since each staff member can work at most two night shifts per week, once a staff member is assigned to Friday in a given week, the second working day is determined by that staff member's fixed evening-shift pattern (e.g. Monday–Tuesday, Tuesday–Wednesday or Wednesday–Thursday). Therefore, assignments for the remaining weekdays can be derived from the Friday decision. This feature enables handling the undesirable days (Friday and Saturday) as a subproblem. Given an optimal solution for this subproblem, all that remains is to complete the schedule with the other coverage constraints.

In regular weeks without holidays, the Friday assignments largely determine the remaining weekday evening duties based on employees' fixed availability patterns and the restriction to at most two duties per week. Moreover, since the fairness objective depends only on Friday and Saturday assignments, the completion step in regular weeks does not alter the objective function's value; it only restores feasibility for the remaining weekdays. In this regard, the proposed decomposition is strongly supported by the problem's structure. In holiday weeks, however, the direct completion may induce additional imbalance in the number of worked days, which motivates the refinement step we introduce in Section 5. Therefore, the decomposition should be understood as a problem-specific structure-exploiting workflow rather than as a formally exact Benders-type decomposition.

Accordingly, we propose a two-stage decomposition approach to solve the problem. In *Stage 1*, we exclusively determine the assignments associated with the undesirable days. Once these rotations are fixed, *Stage 2* completes the schedule for the remaining weekdays, enforcing labour constraints and individual preferences. This approach enables the subproblem to account for the distribution of days off induced by holidays, yielding fairer schedules.

4.1 *Stage 1*: Friday and Saturday rotations

Stage 1, here denoted as *LRM-S*, is a simplified version of *LRM*, with the following modifications:

- Decision variables x_{id} are defined only for Fridays and Saturdays ($x_{id} \in \{0,1\},\ i \in ST,\ d \in NF \cup NS$).
- Constraints (3) are enforced for all $d \in (NF \cup NS) \setminus (\overline{D} \cup \overline{D})$.
- Constraints (6)–(8) are removed.

4.2 *Stage 2*: Weekday evening assignments

Once the Friday and Saturday rotation is defined, the roster for the remaining weekdays must be completed. This assignment is straightforward for weeks with-

out holidays, since each staff member works exactly two days per week. Figure 1 illustrates how the second stage can be completed for a fixed Friday rotation.

Fig. 1: Example of weekday evening-shift completion after fixing the Friday rotation.

Staff	Mo	Tu	We	Th	Fr
1	✓	✓			✓
2	✓	✓			✓
3		✓	✓		✓
4			✓	✓	✓
5			✓	✓	✓

(a) Availability pattern for each staff member.

Staff	Mo	Tu	We	Th	Fr
1					
2					✓
3					
4					
5					✓

(b) Friday assignments fixed in Stage 1.

Staff	Mo	Tu	We	Th	Fr
1	✓	✓			
2					✓
3		✓	✓		
4			✓	✓	
5					✓

(c) Automatic assignment of staff not selected for Friday.

Staff	Mo	Tu	We	Th	Fr
1	✓	✓			
2	✓				✓
3		✓	✓		
4			✓	✓	
5				✓	✓

(d) Completion of the remaining assignments in Stage 2.

Figure 1a presents the weekly availability patterns of the five staff members. In this example, staff members 1 and 2 follow pattern FN_1 and can work on Mondays (Mo), Tuesdays (Tu) and Fridays (Fr); staff member 3 follows pattern FN_2 and can work on Tuesdays, Wednesdays (We) and Fridays; and staff members 4 and 5 follow FN_3 and can work on Wednesdays, Thursdays (Th) and Fridays. Figure 1b shows the Friday assignments determined in Stage 1, highlighted in blue for staff members 2 and 5. Given this Friday decision, the remaining weekday assignments can be inferred from the availability pattern. Figure 1c indicates that staff members who are not assigned on Friday must work on the other two days of their pattern; therefore, staff members 1 and 3 are assigned to Tuesday, while staff member 4 is assigned to Wednesday and Thursday. Finally, Figure 1d shows the completion of the schedule: the staff members assigned to Friday are allocated to the only remaining day in their pattern that still requires coverage. Staff member 2 is assigned to Monday, and staff member 5 to Thursday, yielding the final roster.

The main advantage of this approach lies in the clarity and flexibility of the resulting model. By explicitly identifying the undesirable days, the formulation can accommodate computational adjustments—such as including or excluding holidays—in a direct manner, without requiring a major reformulation of the constraints. Moreover, separating the definition of undesirable-day assignments from the completion of the remaining weekdays significantly reduces the computational time and improves scalability to longer planning horizons. Hence, the proposed approach preserves the key principles of the original model, particu-

larly the pursuit of equity and preference satisfaction, while offering a leaner, modular formulation better suited to practical applications and future analyses.

5 Holiday-week refinement

While solving real-world instances, we observed that weeks containing holidays introduce additional imbalance in the workload distribution, particularly regarding days off induced by non-working days. To address this issue, we developed a complementary model that balances the distribution of working days across weeks with holidays, accounting for different feasible weekly patterns and respecting the maximum number of evening-shifts per week. The formulation seeks to minimise the gap between the maximum and minimum number of worked days across staff members over the planning horizon. In this improvement model, Monday and Thursday in holiday weeks are parameters (Mo_s, Th_s) in addition to Friday assignments (Fr_{is}). The decisions focus on distributing Tuesday/Wednesday assignments so that holiday-related time off is spread as evenly as possible across staff. The refinement model for weeks with holidays, which aims to reduce disparities in the number of worked days (i.e., to minimise the range between the most- and least-loaded staff members), is called *(LRM-H)* and defined by (22)–(36). The corresponding sets, parameters, and decision variables are given in Table 2.

Table 2: Sets, parameters and variables of the model *(LRM-H)*.

Sets	
FN_1	set of staff members with working pattern is Mon., Tue. and Fri.;
FN_2	set of staff members with working pattern is Tue., Wed. and Fri.;
FN_3	set of staff members with working pattern is Wed., Thu. and Fri.;
NSF	set of weeks containing at least one holiday;
Parameters	
Mo_s (Tu_s, We_s, Th_s)	equals 1 if Monday (Tuesday, Wednesday, Thursday) in week $s \in NSF$ is a working day (not a holiday), and 0 otherwise;
Fr_{is}	equals 1 if staff member $i \in ST$ was assigned on Friday of week $s \in NSF$ (and 0 otherwise); if Friday is a holiday, all $Fr_{is} = 0$.
Variables	
t_{is}	is equal to 1 if staff member $i \in FN_1 \cup FN_2$ works on Tuesday in week $s \in NSF$, and 0 otherwise;
q_{is}	is equal to 1 if staff member $i \in FN_2 \cup FN_3$ works on Wednesday in week $s \in NSF$, and 0 otherwise;
$Z_{\min}$ ($Z_{\max}$)	minimum (maximum) number of Monday-Friday assignments across staff members over holiday weeks.

$$\textit{Minimise} \quad Z_{\max} - Z_{\min} \tag{22}$$

subject to

$$Z_{\max} \geq \sum_{s \in NSF} (Mo_s + t_{is} + Fr_{is}) \qquad i \in FN_1 \tag{23}$$

$$Z_{\max} \geq \sum_{s \in NSF} (t_{is} + q_{is} + Fr_{is}) \qquad i \in FN_2 \tag{24}$$

$$Z_{\max} \geq \sum_{s \in NSF} (q_{is} + Th_s + Fr_{is}) \qquad i \in FN_3 \tag{25}$$

$$Z_{\min} \leq \sum_{s \in NSF} (Mo_s + t_{is} + Fr_{is}) \qquad i \in FN_1 \tag{26}$$

$$Z_{\min} \leq \sum_{s \in NSF} (t_{is} + q_{is} + Fr_{is}) \qquad i \in FN_2 \tag{27}$$

$$Z_{\min} \leq \sum_{s \in NSF} (q_{is} + Th_s + Fr_{is}) \qquad i \in FN_3 \tag{28}$$

$$Mo_s + t_{is} + Fr_{is} \leq 2 \qquad i \in FN_1,\ s \in NSF \tag{29}$$

$$t_{is} + q_{is} + Fr_{is} \leq 2 \qquad i \in FN_2,\ s \in NSF \tag{30}$$

$$q_{is} + Th_s + Fr_{is} \leq 2 \qquad i \in FN_3,\ s \in NSF \tag{31}$$

$$\sum_{i \in FN_1 \cup FN_2} t_{is} = 2\ Tu_s \qquad s \in NSF \tag{32}$$

$$\sum_{i \in FN_2 \cup FN_3} q_{is} = 2\ We_s \qquad s \in NSF \tag{33}$$

$$Z_{\min}, Z_{\max} \geq 0 \tag{34}$$

$$t_{is} \in \{0, 1\} \qquad i \in FN_1 \cup FN_2,\ s \in NSF \tag{35}$$

$$q_{is} \in \{0, 1\} \qquad i \in FN_2 \cup FN_3,\ s \in NSF \tag{36}$$

Objective (22) minimises the range of evening assignments across staff members in holiday weeks, promoting equity. Constraints (23)–(25) and (26)–(28) define upper and lower bounds on each staff member's cumulative number of weekday assignments over the weeks with holidays. For a given staff member, this cumulative workload consists of the weekday assignments that are fixed once the holiday calendar is known (e.g. the parameter Mo_s is equal to zero on holidays), together with the Tuesday/Wednesday assignments determined by the refinement model (t_{is} and q_{is}) and the Friday assignment Fr_{is} fixed in Stage 1. Since this expression depends on the employee's weekly availability pattern, three pattern-specific pairs of constraints are required. In this way, $Z_{\min}$ and $Z_{\max}$ represent, respectively, the smallest and largest cumulative workloads among all staff members in holiday weeks. Analogously, Constraints (29)–(31) ensure that the total number of assignments received by a staff member in a holiday week does not exceed two. Finally, Constraints (32)–(33) enforce two assignees on Tuesday/Wednesday when these are not holidays. The variable domains are defined by Constraints (34)–(36).

6 Computational results

This section reports results from applying the developed decomposition approach to real planning instances from the ICMC/USP Library. More precisely, the semester rosters were first generated by applying Stage 1 to determine Friday and Saturday assignments and Stage 2 to complete the remaining weekday evening duties. The holiday-week refinement model serves only to evaluate its effect on workload equity in weeks affected by holidays.

We applied the two-stage decomposition approach to four semester-long rostering instances (one instance per semester), covering the second semester of 2024

through the first semester of 2026. Each semester was solved separately. The instances were solved using the SCIP Optimization Suite (10.0.1) on a computer with an Intel(R) Core(TM) i5-10300H processor, 16 GB of RAM, and the Windows 11 operating system. Across the four semesters, the Stage 1 models comprised, on average, 172 decision variables and 558 constraints, while the Stage 2 models comprised 110 decision variables and 143 constraints. The computational time required to generate the schedule for each semester was less than one second. Across these four instances, the minimum normalised happiness achieved was at least 0.95, ensuring that even the least satisfied staff member attained at least 95% of their maximum attainable happiness. These results show consistently high and balanced preference satisfaction regarding undesirable duties, despite the seasonal variability in the number and distribution of holidays and in the number and timing of undesirable-day assignments that had to be covered.

Figure 2 reports the effect of the holiday-week refinement on workload equity. The bars labelled "Without Holidays" correspond to the rosters obtained solely from the two-stage decomposition. In contrast, the bars labelled "With Holidays" correspond to the rosters after applying the holiday-week refinement model. The results indicate that incorporating this refinement yields a more equitable distribution of days off, thereby reducing the difference over the planning period and improving the perceived fairness of the resulting roster.

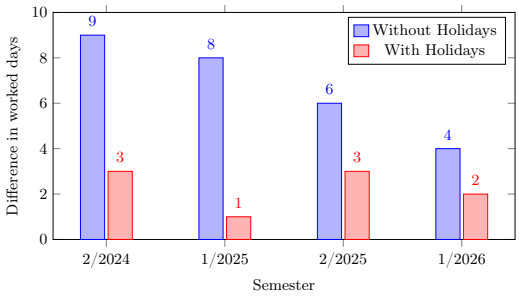


Fig. 2: Difference between the most- and least-loaded staff members in terms of worked days over the semester.

7 Final remarks

This paper presents a practical, optimization-based approach to generating staff schedules for a public university library. The proposed method combines a compact MILP formulation, a two-stage decomposition focused on undesirable duties, and a refinement step for weeks with holidays. In contrast to the manual, time-consuming roster construction previously performed by library staff, the developed approach generates high-quality schedules rapidly and transparently, facilitating its routine use in practice. Across four consecutive semesters, the minimum normalised satisfaction among staff members remained high (95%), indicating consistently robust preference satisfaction despite the variability in holiday configurations and operational conditions. In conclusion, the generated

schedules were presented to the library team and have been successfully integrated into their routine since the second semester of 2025, contributing to greater perceived fairness and transparency in the rostering process.

Future research may explore the performance of the developed models and solution framework for larger instances, as well as the incorporation of additional operational constraints common to library settings. It would be interesting to investigate whether the personnel scheduling problem addressed in this paper can be solved with straightforward algorithms without using MILP solvers.

Acknowledgements This work was funded by the São Paulo Research Foundation (FAPESP), Brazil, grants #2025/01587-2 and #2023/16405-1. The authors also acknowledge financial support from the National Council for Scientific and Technological Development (CNPq) grant #309161/2022-3, and the Research Foundation – Flanders (FWO grant K804824N).

References

1. Ásgeirsson, E.I.: Bridging the gap between self schedules and feasible schedules in staff scheduling. Annals of Operations Research **218**, 51–69 (2014)
2. Burke, E.K., De Causmaecker, P., Vanden Berghe, G., Van Landeghem, H.: The state of the art of nurse rostering. Journal of Scheduling **7**, 441–499 (2004)
3. Ceschia, S., Di Gaspero, L., Schaerf, A.: Educational timetabling: Problems, benchmarks, and state-of-the-art results. European Journal of Operational Research **308**(1), 1–18 (2023)
4. De Causmaecker, P., Vanden Berghe, G.: A categorisation of nurse rostering problems. Journal of Scheduling **14**(1), 3–16 (2011)
5. Ernst, A., Jiang, H., Krishnamoorthy, M., Sier, D.: Staff scheduling and rostering: A review of applications, methods and models. European Journal of Operational Research **153**(1), 3–27 (2004)
6. Fuchs, G., Schimmelpfeng, K., Brunner, J.O.: A roadmap for integrating fairness in personnel planning and scheduling in hospitals. Operations Research, Data Analytics and Logistics **45**, 200479 (2025)
7. Legrain, A., Omer, J.: A dedicated pricing algorithm to solve a large family of nurse scheduling problems with branch-and-price. INFORMS Journal on Computing **36**(4), 1108–1128 (2024)
8. Rocha, M., Oliveira, J.F., Carravilla, M.A.: Cyclic staff scheduling: optimization models for some real-life problems. Journal of Scheduling **16**, 231–242 (2013)
9. Santos, G.H., Toffolo, A.T., Gomes, A.R.: Integer programming techniques for the nurse rostering problem. Annals of Operations Research **239**, 225–251 (2016)
10. Smet, P., Brucker, P., De Causmaecker, P., Vanden Berghe, G.: Polynomially solvable personnel rostering problems. European Journal of Operational Research **249**(1), 67–75 (2016)
11. Valouxis, C., Gogos, C., Goulas, G., Alefragis, P., Housos, E.: A systematic two phase approach for the nurse rostering problem. European Journal of Operational Research **219**(2), 425–433 (2012)
12. Wickert, T.I., Kummer Neto, A.F., Boniatti, M.M., Buriol, L.S.: An integer programming approach for the physician rostering problem. Annals of Operations Research **302**, 363–390 (2021)

Integrative Hybrid Local Search with a Restart Strategy for University Course Timetabling with Student Sectioning*

Mingxuan Li[1][0009-0006-9735-2900], Thomas Weise[2][0000-0002-9687-8509], Sina Abdipoor[3**][0000-0001-8411-5451], Jourdan D'orville[4][0000-0002-2060-5758], Say Leng Goh[5][0000-0003-4562-2519], Razali Yaakob[6][0000-0002-8228-5753], and Salwani Abdullah[7][0000-0003-0037-841X]

[1] BUPT International School, Queen Mary University of London, London E1 4NS, United Kingdom
mingxuan.li@se23.qmul.ac.uk
[2] School of Artificial Intelligence and Big Data, Hefei University, Hefei, Anhui, China
tweise@hfuu.edu.cn
[3] Department of Data Science and Artificial Intelligence, Faculty of Engineering and Technology, Sunway University, 47500 Selangor, Malaysia
sinaa@sunway.edu.my
[4] Department of Actuarial Science and Risk, School of Mathematical Sciences, Sunway University, 47500 Selangor, Malaysia
jourdand@sunway.edu.my
[5] Optimization and Visual Analytics Research Group, Faculty of Computing and Informatics, Universiti Malaysia Sabah, 87000 Labuan, Malaysia
slgoh@ums.edu.my
[6] Department of Computer Science, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 UPM Serdang, Malaysia
razaliy@upm.edu.my
[7] Data Mining and Optimization Lab, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 UKM Bangi, Selangor, Malaysia
salwani@ukm.edu.my

Abstract. The optimal balance between exploitation and exploratory capabilities of optimization algorithms plays a vital role in convergence on combinatorial optimization problems. While hybridizing diversification and intensification has been proven effective in scheduling tasks, the mechanism for switching between them during different stages of the search remains less explored. In this study, a restart strategy based on Luby's universal sequence is adopted as a neighborhood structure guidance to manage the transition between exploration and exploitation,

* Supported by Universiti Kebangsaan Malaysia.
** Corresponding author.

guided by theoretical indications of its efficacy. To validate this, a diversification approach based on Random Search is hybridized with an intensification approach based on Hill Climbing. This methodology is applied to a university course timetabling with student sectioning benchmark as a proof of concept. When tested on the International Timetabling Competition 2019 dataset, the empirical results align with theoretical expectations and outperform the individual building blocks at a faster pace, finding 80% more feasible candidate solutions. Beyond this experiment, these findings are significant for balancing search algorithms within Operations Research in general and contribute to other optimization tasks.

Keywords: Operations Research · Luby's Restart Strategy · Explore/-Exploit Balance · University Course Timetabling · International Timetabling Competition (ITC) 2019.

1 Introduction

The University Course Timetabling Problem (UCTP) [5,2] is a major branch of Educational Timetabling Problems (ETPs) that has attracted significant attention in recent years. This Combinatorial Optimization Problem (COP) is a special case of Graph Coloring with many real-world applications [11]. The UCTP is an $\mathcal{NP}$-hard problem, and the worst-case runtime to solve it grows exponentially with the problem scale [6]. This scheduling task must be performed repeatedly before each academic year. Since manual timetabling is often infeasible [12], an efficient automated solver is vital in both theory and practice.

While significant advancements have been made on the UCTP, a major gap remains in the adoption of these methods in real-world applications. To bridge this gap, several practical aspects were incorporated into the International Timetabling Competition (ITC-2019) dataset[8] [20], which remains the latest ETP benchmark. The current state-of-the-art (SOTA) approaches are primarily based on Mathematical and Matheuristic methods. Despite their strong performance, they may require more resources than metaheuristics [33,26,9] and also tend to be much more complicated.

Black-box optimization methods typically make very few assumptions about the nature of the problem and can easily be adapted to new domains. They usually are general stochastic Monte Carlo methods and tend to search the space of possible solutions more efficiently than deterministic methods [3]. Such algorithms are able to achieve decent results within short time frames across many COPs [27]. The two simplest black-box metaheuristics are Hill Climbing (HC) and Random Search (RS). The former puts all search effort into exploitation. The latter puts all search effort into exploration. Balancing these two aspects remains a significant open problem in metaheuristics [25,7,8,31].

This work proposes a hybrid local search approach based on RS and HC to achieve feasibility on the ITC-2019 benchmark. It incorporates the universal

[8] https://www.itc2019.org Last accessed: 19 March 2026

strategy from [18], which we refer to as *Luby's Restart Strategy*, to balance exploration and exploitation. It furthermore utilizes a Distance to Feasibility (DF) [1] neighborhood guidance strategy. Based on theoretical studies of the first hitting time, the potential for improved feasibility convergence in the Hybrid RS and HC with Luby's Restart Strategy (HRSHCL) is established. When applied and tested on the ITC-2019 dataset, the empirical results reinforce these theoretical claims. Beyond outperforming the individual RS and HC components and achieving 80% higher feasibility, a major contribution of this study is the investigation of Luby's Restart Strategy as a balance strategy in Monte Carlo methods. This approach offers significant potential for other hybrid metaheuristics and various COPs. With this work, we make four main contributions:

1. We propose a novel restart strategy combining a Monte Carlo hill climber (HC) and random search (RS) based on theories on Las Vegas algorithms.
2. We show that this method can increase the number of feasible solutions discovered compared to the two-component algorithms on ITC-2019.
3. We provide a theoretical foundation proving that this was the likely outcome of our experiments, i.e., why HRSHCL is better than both HC and RS.
4. We provide an open-source implementation of all algorithms in a GitHub repository https://github.com/ScholORs/itc-2019 for replication.

2 University Course Timetabling with Student Sectioning

The University Course Timetabling with Student Sectioning Problem (UCTSSP) [20] is a combination of the UCTP [5] with student sectioning [15].

2.1 Problem Definition

The UCTSSP consists of two primary components [21]: (1) allocating available times and rooms to each class and (2) assigning students to those classes. These assignments must satisfy a specific set of predefined constraints. Assigning students to specific sections is an essential aspect of real-world problems, which has been ignored in previous UCTPs. The UCTSSP, therefore, represents the actual needs of schedulers. However, it also has a significantly larger search space, which increases the problem difficulty exponentially [16].

2.2 The International Timetabling Competition 2019

The ITC-2019 benchmark [20] consists of 30 problem instances of different sizes and attributes. Formally, an instance is defined by four entities: times, rooms, classes, and students. A detailed description of the dataset is provided in [21].

To evaluate a generated schedule, the distribution constraints are bifurcated into two functional categories: (1) hard constraints that dictate schedule feasibility, and (2) soft constraints that determine the overall optimization quality. From this perspective, one may treat the ITC-2019 as a bi-objective problem. The first objective measures the proximity of a timetable to a feasible state. The second objective encompasses the cost of the solution.

The Distance to Feasibility Metric $DF : \mathcal{S} \mapsto \mathbb{N}_0$ is the sum of penalties due to hard constraint violations, rooms assigned to multiple classes with overlapping times, and classes that are assigned unavailable rooms. As the main objective of this work is to achieve feasible scheduled classes, the student sectioning subproblem is dismissed by the *DF* metric. Let *Hard*, *Room*, and *Class* denote the set of all hard constraints, rooms, and classes, respectively. For each hard constraint H, if a timetable TT violates H, then $H(TT) = 1$. Otherwise, $H(TT) = 0$. For each room r, let $Overlap(r)$ count the number of additional classes assigned to r within an overlapping time period. For each class c with an assigned time and room, let $BadRoom(c)$ be 1 if the room assigned to c is unavailable at the assigned time. Otherwise, $BadRoom(c) = 0$. Then DF is subject to minimization and is defined as follows [1]:

$$DF(TT) = \sum_{H \in Hard} H(TT) + \sum_{r \in Room} Overlap(r) + \sum_{c \in Class} BadRoom(c). \tag{1}$$

A candidate solution TT is *feasible* if and only if $DF(TT) = 0$. It has been shown that the most significant challenge of the ITC-2019 benchmark is achieving feasibility [28,19,24]. We therefore focus on the DF objective in our work.

Cost Metric The *Cost* of a solution TT is the weighted sum of penalties incurred by its soft constraint violations. It is formulated as follows:

$$Cost(TT) = W_t \sum_{i=1}^{a} T_{c_i} + W_r \sum_{i=1}^{a} R_{c_i} + W_d \sum_{j=1}^{b} D_j + W_s \sum_{k=1}^{c} SC_{s_k} \tag{2}$$

W_t, W_r, W_s, W_d denote the weights for assigned time (T_{c_i}), room (R_{c_i}), distribution constraint (D_j), and student sectioning (SC_{s_k}) violations, respectively.

We approach the UTCSSP in two steps. Two solutions are first compared based on their distance to feasibility DF. Only when comparing two feasible solutions do our algorithms use the cost function *Cost* as a tie-breaker. This can be implemented by minimizing the combined objective function f, where $UB(Cost)$ is the upper bound of the cost function, which, in a practical implementation, can be replaced by a reasonably large integer constant.

$$f(TT) = \begin{cases} UB(Cost) + DF(TT) \text{ if } DF(TT) > 0 \\ Cost(TT) \text{ otherwise} \end{cases} \tag{3}$$

3 Related Works

Metaheuristics [29,4] are widely applied to COPs in the literature. These approaches dominate both the theory and practice of ETPs, with the current SOTA for most UCTP benchmarks being metaheuristic-based [5]. Hybrid metaheuristics are particularly effective as they efficiently balance exploration and exploitation. They are generally categorized into integrative and collaborative [2]. While integrative approaches embed one method within the mechanics of another, collaborative methods apply techniques sequentially [23].

Restarts are well-established techniques that involve running an algorithm with certain inputs for certain iterations before restarting it with a new set of inputs [29]. For an overview of the success of restarts in various settings, see [22]. From a simplified perspective, assume that an algorithm $\mathcal{A}$ may yield different results if executed several times on a problem instance $\mathcal{I}$. If these results have different objective values, some of them are necessarily better. Therefore, one could execute the same algorithm several times, each time using the original computational budget $\mathcal{B}$, and retain the best result discovered. In this case, the overall best is better than that of a single execution with a high probability. Restarting thus utilizes the *variance* of the results [29]. However, it is only efficient if we can divide the original budget $\mathcal{B}$ into several shorter slices $b_1 + b_2 + \cdots + b_z = \mathcal{B}$ and still stochastically have a higher chance of getting better results when restarting the algorithm $\mathcal{A}$ using only these slices. This raises the non-trivial questions of how many such slices we should have (here: the number z) and how long they should be. For discrete time processes, one may apply the universal strategy proposed by Luby, Sinclair, and Zuckerman [18] who proved their strategy is optimal for Las Vegas algorithms. In contrast, Lorenz [17] showed that the continuous setting admits no optimal strategy. For our context, we schedule restarts according to Luby's Restart Strategy to balance exploratory and exploitative elements of the proposed hybrid algorithm. The details are given in Section 4.

In the context of the UCTSSP and the ITC-2019 benchmark specifically, metaheuristics offer significant potential for further exploration [3]. Most methods applied during and post the competition were mathematical or matheuristic-based [19,13,20,24,15], with Simulated Annealing [28] and Forest Growth [10] being notable exceptions. While these approaches represent the current SOTA, they are highly resource-intensive, with some instances requiring execution times exceeding 10 days [24]. Such runtimes are impractical and often infeasible in real-world settings [3]. Our proposed integrative hybrid methodology represents a novel application for this problem, bridging the existing gap in the literature.

4 Proposed Methodology

We now define a local search methodology that avoids premature convergence to local optima and strikes a balance between exploitation and exploration. We demonstrate how our integrative local search with the Luby strategy can produce more feasible solutions on the UCTSSP.

4.1 Random Search and Hill Climbing

In our previous research, we tested the two extreme ends of the "algorithm spectrum" regarding exploration and exploitation. On the exploration end, we used Random Search (RS, Algorithm 1), which creates a new random timetable TT_n in each step, not making use of any previously discovered information, and returns the timetable TT_b with the smallest objective value y_b it encountered. This algorithm, hence, was not able to find any feasible solution in our experiments.

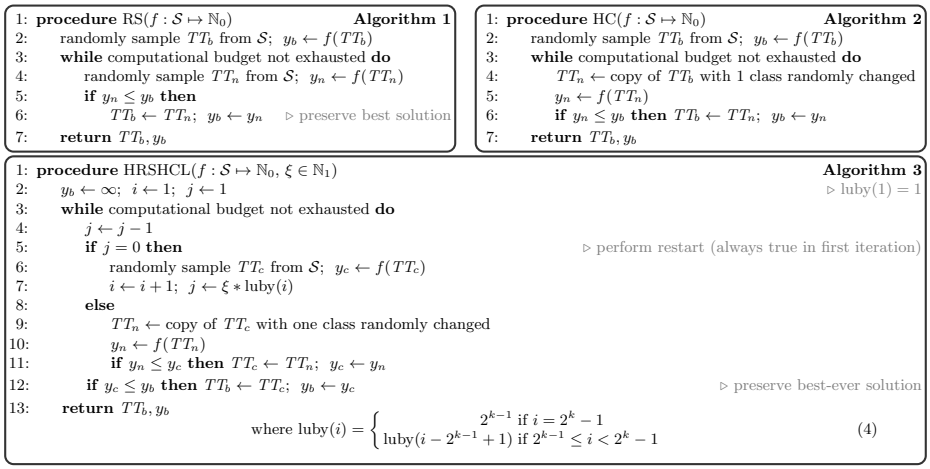


Fig. 1. The three algorithms compared in our study using the objective function f combining the distance-to-feasibility *DF* and the cost objective *Cost* (as given in Equation 3) and returning the best timetable TT_b discovered and with its objective value y_b.

On the exploitation end of the scale, we used Hill Climbing (HC, Algorithm 2). It, too, starts with a random timetable TT_b. In each step, it creates a modified copy TT_n of TT_b by changing one randomly selected class assignment. A neighborhood move consists of randomly selecting a class and, with equal probability, either assigning it to a new valid time slot or a new valid room (if that class requires a room). If this new timetable TT_n is not worse than TT_b, i.e., if its objective value y_n is less than or equal to the objective value y_b of the best-so-far solution TT_b, it replaces TT_b. This algorithm focuses entirely on exploitation and cannot escape a local optimum. HC indeed found feasible solutions in our experiments and its average *DF* was much lower than that of RS. However, the fact that it was easily trapped in local optima proved problematic. In our previous experiment, we demonstrated that on many instances of the ITC-2019, HC was susceptible to getting stuck in local optima before exhausting the runtime budget [1].

Restarting the algorithm while storing the best-found solution in a separate variable, therefore, appears to be a promising strategy.

4.2 Restart Strategy

In Algorithm 3, we present the HRSHCL which combines the hill climber HC with the restart strategy proposed by Luby, Sinclair, and Zuckerman [18], here called the "Luby Restart Strategy." One advantage of this is that it performs runs of different durations with the local search and therefore requires less configuration effort. The hill climber is embedded in HRSHCL iteratively to refine the timetable TT_c by sampling new solutions TT_n from its neighborhood. Exactly as in HC, if they are not worse than TT_c, they replace it. However,

the hill climber only gets to perform $\xi * \text{luby}(i)$ FEs, where i is the index of the current restart, $\xi \in \mathbb{N}_1$ is a configuration parameter, and $\text{luby}(i)$ is the i-th element from the universal sequence (Equation 4 inside Algorithm 3) and [18]. In other words, the number of steps until restart follows the sequence $(1, 1, 2, 1, 1, 2, 4, 1, 1, 2, 1, 1, 2, 4, 8, 1, \dots)$ multiplied by ξ.

Whenever this internal budget is exhausted, TT_c is randomly sampled from the search space $\mathcal{S}$, exactly in the same way that RS does. Of course, the best-ever encountered timetable TT_b is retained in an additional variable and returned after the computational budget is exhausted (together with its objective value y_b).

The Luby Restart Strategy is optimal for Las Vegas algorithms. Las Vegas algorithms always produce the right solution if they terminate, but their runtime is a random variable. The HC algorithm most likely is not a Las Vegas algorithm. Based on experience and results from literature on other domains [31,30], we *assume* that it gets trapped in local optima. While there is no proof of that, we can indeed observe the fact that all of the HC runs stop improving after a certain amount of time. This does not mean that they would not eventually improve. After all, it could be that the optimal solution in $\mathcal{S}$ might be connected to the current solution of the HC run via a very large neutral network. Therefore, at least for the case that HC could actually be a Las Vegas strategy under our choice of $\mathcal{S}$ and DF, the Luby Restart Strategy would be the *ideal* restart strategy.

Fewer results are known for the optimization case [32]. In the more likely case that HC is not a Las Vegas algorithm in our scenario, however, the Luby Restart Strategy has the striking advantage that it requires very little configuration. Determining the optimal step count for a restart is vital for performance, as different instances require different configurations. The Luby Restart Strategy facilitates this by assigning varying budget lengths to each restart sequence.

4.3 Theoretical Motivation of HRSHCL

Let *NFE* be the *number of fitness evaluations* and let *NFEF* be the *number of fitness evaluations to feasibility*. That is, *NFEF* is the *NFE* needed to first reach $DF = 0$, or in other words, the first hitting time.

We can define the problem as the pair $(\mathcal{S}, DF)$, where the search space $\mathcal{S}$ contains all possible timetables, and the objective function DF rates them. Each timetable TT contains classes that have been assigned a time and a room, regardless of the assignment's feasibility. The set of global optima $\mathcal{G} \subseteq \mathcal{S}$ contains all timetables that satisfy all hard constraints and the maximum number of soft constraints. Hence, if $TT \in \mathcal{G}$, then $DF(TT) = 0$ and $Cost(TT)$ is the smallest possible over all feasible $TT \in \mathcal{S}$. For any TT, let $N(TT)$ be the neighborhood of TT, which is the set containing all timetables that can be reached in exactly one search step. In our case, this means that they differ from TT by exactly 1 class assignment. A local optimum is a solution TT whose neighborhood is comprised entirely by inferior solutions, i.e., such that HC would never select.

Let $p_{\text{RS}} = \mathbb{P}(\mathcal{G})$ be the probability that the RS algorithm randomly samples a global solution (based on the distribution induced by the algorithm). Since

each RS iteration independently randomly samples a TT from $\mathcal{S}$, multiple iterations of RS form a sequence of i.i.d. Bernoulli trials. Hence, the the first hitting time $NFEF_{\mathrm{RS}}$ of RS is a geometric random variable with expectation

$$\mathbb{E}(NFEF_{\mathrm{RS}}) = \frac{1}{p_{\mathrm{RS}}}. \tag{5}$$

Let $TT_1 \in \mathcal{S}$ be some timetable sampled by the RS algorithm with probability $\mathbb{P}(TT_1)$. Let $TT_2 \in \mathcal{S}$ be the best HC solution during its course. Finally, $\mathbb{P}(TT_2 \mid TT_1)$ be the probability that TT_2 is reached if the starting point of HC is TT_1. HC is successful if $TT_2 \in \mathcal{G}$ and its probability p_{HC} of success is

$$p_{\mathrm{HC}} = \sum_{TT_1 \in \mathcal{S}} (\mathbb{P}(TT_1) \times \mathbb{P}(TT_2 \mid TT_1)). \tag{6}$$

Since each restart from HRSHCL consists of an initial TT generated by RS followed by iterations of HC up to the budget set by the Luby method, assuming global optima is reached within this budget, then the probability of success from a single restart of HRSHCL is also p_{HC}.

Let $\mathbb{E}(NFEF_{\mathrm{HRSHCL}})$ be the expected $NFEF$ for HRSHCL. On average, each restart will have $\mathbb{E}(NFE_{\mathrm{HC}})$ many DF evaluations. Next, similar to RS, we may consider each restart in HRSHCL as an i.i.d. Bernoulli trial with success probability p_{HC}. So, the expected number of restarts to the first success is $1/p_{\mathrm{HC}}$. It follows that the expected $NFEF_{\mathrm{HRSHCL}}$ is

$$\mathbb{E}(NFEF_{\mathrm{HRSHCL}}) = \mathbb{E}(NFE_{\mathrm{HC}}) \times \frac{1}{p_{\mathrm{HC}}}. \tag{7}$$

The exploitation element of HC in HRSHCL reaches a globally optimal solution from many more points compared to RS. So $p_{\mathrm{HC}} > p_{\mathrm{RS}}$. Furthermore, we may assume that $\mathbb{E}(NFE_{\mathrm{HC}}) < 1/p_{\mathrm{RS}}$ because p_{RS} is very small. Hence,

$$\mathbb{E}(NFEF_{\mathrm{HRSHCL}}) = \frac{\mathbb{E}(NFE_{\mathrm{HC}})}{p_{\mathrm{HC}}} < \frac{1}{p_{\mathrm{RS}}} = \mathbb{E}(NFEF_{\mathrm{RS}}). \tag{8}$$

Multiple integrative restarts introduce an exploratory aspect that offsets the fully exploitative nature of HC, enabling HRSHCL to escape local optima. Note that HC alone has a single attempt at success with probability p_{HC}. In contrast, we require at least 1 success from k restarts of HRSHCL, with probability

$$1 - (1 - p_{\mathrm{HC}})^k. \tag{9}$$

As the number k of restarts increases, the probability of success approaches 1. While HC is not guaranteed to reach a global solution, HRSHCL eventually does. Therefore, HRSHCL theoretically performs better than RS or HC alone.

5 Empirical Results

We now analyze the performance of the proposed HRSHCL against its individual building blocks (RS and HC) on the ITC-2019 benchmark. While our approach does not yet reach the performance of specialized SOTA methods like Simulated Annealing [28], which achieves feasibility across all instances, the primary focus of this empirical study is to demonstrate how a simple restart strategy can significantly enhance the performance of basic metaheuristic components as a proof of concept.

5.1 Experimental Setup

All empirical experiments were conducted under identical conditions as summarized in Table 1. Compared to the high-performance testbeds often cited in the literature [20], our hardware reflects the computing power typically available to educational institutions in real-world applications. Each problem instance was assigned a 4-hour time budget. Compared with the current SOTA, which can require more than 10 days on each problem instance with high-performance computers [24], our experiment offers a practical, lightweight constraint model representative of the computing power available to most educational institutions. Despite this more modest time budget, a full execution on the entire dataset requires 120 hours. This highlights the complexity of the benchmark. The only algorithm requiring parameterization is HRSHCL, for which we uniformly set $\xi = 100,000$ across all instances.

Table 1. Empirical Experiment Setup

Operating System	Windows Server 2019 Standard
Processor	Intel(R) Xeon(R) E-2314, 4 cores, 2.80 GHz
RAM	16 GB DDR4, 3200 MHz (DIMM)
Instance Count	30 problem instances
Runtime per Run	4 hours per instance
Total Runtime	120 hours per full dataset run

5.2 Comparative Distance to Feasibility Analysis

Table 2 compares the performance of the three algorithms on the 30 instances of the ITC-2019 benchmark. The evaluation focuses on achieving feasibility. We utilize the *NFE* as our primary metric to ensure machine-independent comparisons of algorithmic efficiency, alongside total runtime

The simple HC clearly outperforms RS, as it can find feasible solutions ($DF = 0$) on five instances. The Luby Restart Strategy in the HRSHCL improves this by discovering 9 feasible solutions. If HC can find a feasible solution on an instance, the HRSHCL can do that, too. More importantly, HRSHCL can find feasible timetables in four significantly harder instances, namely mary-spr17, muni-fsps-spr17c, nbi-spr18, and lums-fal17. The HRSHCL can escape local optima that trap HC.

However, there is a tradeoff. On several instances, the best DF achieved by HRSHCL is higher than that of HC. This indicates that our uniform setting of ξ may not be the optimal choice across all instances. Different instances have different scales, and the larger the scale, the larger the neighborhoods around the solutions. Hence, the time until a hill climber gets trapped is drastically different. We still chose to retain our static $\xi = 100,000$ setting for fairness across the compared algorithms. In our future work, we will set ξ adaptively.

Table 2. Performance comparison of RS vs. HC vs. HRSHCL in four hour runs, based on total FEs *NFE*, feasibility (*fs*?), and the first hitting time *NFEF*.

		4-Hour Experiments												12-Hour Experiment			
Instance		Random Search (RS)				Hill Climber (HC)				HRSHCL (First 4h)				HRSHCL (Full)			
#	Name	*fs?*	*NFE*	*NFEF*	*DF*	*fs?*	*NFE*	*NFEF*	*DF*	*fs?*	*NFE*	*NFEF*	*DF*	*fs?*	*NFE*	*NFEF*	*DF*
1	agh-fis-spr17	✗	5142450	-	538	✓	15912844	1268040	0	✓	19191088	7788896	0	✓	38362389	7788896	0
2	agh-ggis-spr17	✗	3266295	-	1437	✗	4505760	-	36	✗	2979623	-	29	✗	2979623	-	29
3	bet-fal17	✗	6682078	-	992	✗	13680907	-	41	✗	18308622	-	41	✗	18308622	-	41
4	iku-fal17	✗	1752870	-	2129	✗	3284413	-	45	✗	3178816	-	69	✗	7842904	-	42
5	mary-spr17	✗	10120732	-	498	✗	10530737	-	2	✓	13596346	285689	0	✓	32789460	285689	0
6	muni-fi-spr16	✗	25361531	-	327	✗	55775479	-	3	✗	469413	-	4	✗	469413	-	4
7	muni-fsps-spr17	✗	26209608	-	226	✗	30775425	-	2	✗	86178	-	3	✗	64312766	-	2
8	muni-pdf-spr16c	✗	1748276	-	1316	✗	3188289	-	21	✗	3134724	-	17	✗	7507744	-	12
9	pu-llr-spr17	✗	14158052	-	467	✓	21155104	605051	0	✓	18898741	384538	0	✓	38371289	384538	0
10	tg-fal17	✗	11873109	-	215	✗	14981192	-	8	✗	2168712	-	9	✗	2168712	-	9
11	agh-ggos-spr17	✗	5625426	-	937	✗	11401606	-	14	✗	7815517	-	7	✗	42124648	-	3
12	agh-h-spr17	✗	7189289	-	265	✗	50184161	-	5	✗	33365204	-	6	✗	85231552	-	5
13	lums-spr18	✗	2487250	-	534	✓	52887926	1120530	0	✓	44140686	1063268	0	✓	44140686	1063268	0
14	muni-fi-spr17	✗	28487751	-	340	✗	34004498	-	15	✗	4462818	-	13	✗	4462818	-	13
15	muni-fsps-spr17c	✗	7848425	-	423	✗	22286141	-	2	✓	18586372	16718538	0	✓	44072512	16718538	0
16	muni-pdf-spr16	✗	2004896	-	863	✗	9153338	-	2	✗	7773956	-	2	✗	7773956	-	2
17	nbi-spr18	✗	15984996	-	484	✗	24599466	-	1	✓	18977385	2576303	0	✓	100915300	2576303	0
18	pu-d5-spr17	✗	11294097	-	587	✗	23273746	-	1	✗	3173653	-	2	✗	3173653	-	2
19	pu-proj-fal19	✗	177028	-	4479	✗	260277	-	582	✗	99944	-	903	✗	399926	-	580
20	yach-fal17	✗	22413440	-	302	✓	53125090	417258	0	✓	44075184	5463582	0	✓	140773607	5463582	0
21	agh-fal17	✗	413694	-	4089	✗	1022449	-	143	✗	1087758	-	298	✗	3194803	-	129
22	bet-spr18	✗	5815554	-	974	✗	13508382	-	51	✗	6381674	-	49	✗	44246634	-	45
23	iku-spr18	✗	1770357	-	2131	✗	3054511	-	59	✗	3196771	-	65	✗	7876593	-	50
24	lums-fal17	✗	2335707	-	643	✗	46712367	-	1	✓	30399097	30067527	0	✓	99771512	30067527	0
25	mary-fal18	✗	13069519	-	418	✓	23248935	123112	0	✓	3194895	360381	0	✓	77571936	360381	0
26	muni-fi-fal17	✗	26612982	-	380	✗	55766681	-	14	✗	67043	-	14	✗	67043	-	14
27	muni-fspsx-fal17	✗	4387325	-	948	✗	6293964	-	18	✗	3063474	-	21	✗	18772120	-	11
28	muni-pdfx-fal17	✗	647737	-	3441	✗	1502582	-	69	✗	1183530	-	101	✗	3176811	-	46
29	pu-d9-fal19	✗	2201343	-	1634	✗	3449822	-	20	✗	3123984	-	32	✗	7852718	-	21
30	tg-spr18	✗	13538228	-	232	✗	20494576	-	13	✗	161867	-	11	✗	46479128	-	10

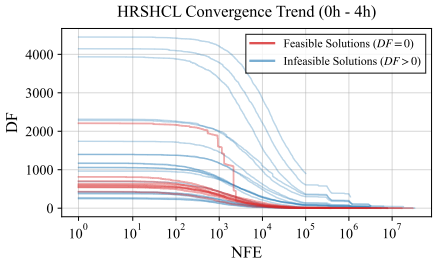


Fig. 2. Distance to Feasibility vs. NFE for HRSHCL (Initial 4 Hours)

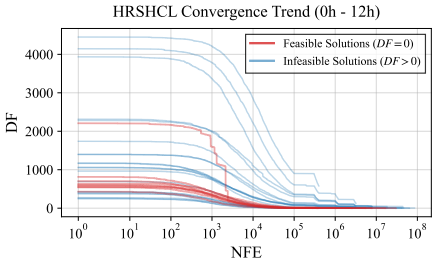


Fig. 3. Distance to Feasibility vs. NFE for HRSHCL (Full 12 Hours)

5.3 Long-Run Performance Analysis

To investigate the performance ceiling of our algorithms, we now expand the computational budget from 4 hours to 12 hours. The empirical results for the extended run of HRSHCL are illustrated in Figure 3 and detailed in Table 2. Compared to the 4-hour runs (Figure 2), the primary steep descent of the *DF* ends within 10^2 to 10^6 *NFE* interval. The overarching curves appear to flatten in the latter hours. However, the examination of the late-stage intervals (the 4h-12h time frame) shows that the progress does not flatline into an absolute dead end. Instead, they exhibit distinct, episodic step-wise descents even after 10^7 *NFE*.

Similarly, the detailed results in Table 2 show that although no additional feasibility was achieved after the 4-hour mark, the *DF* continued to decrease over the subsequent 8 hours. Specifically, instances with a higher *DF* at the 4-hour mark showed continued improvement. This further demonstrates the effectiveness of the Luby restart strategy in facilitating exploration and helping the algorithm escape local optima.

This shift highlights the rugged UCTSSP landscape and the utility of the Luby Restart Strategy. Facing tightly coupled constraints, the restarts warp the search away from local optima. It then completely rebuilds the solution structure, which costs time but eventually leads to late-stage improvements. Ultimately, the 12-hour analysis shows robustness of HRSHCL to resolve complex constraints.

6 Conclusion and Future Work

With this work, we have significantly extended the body of knowledge on the University Course Timetabling with Student Sectioning Problem. The proposed HRSHCL algorithm can find feasible solutions on 80% more instances (9 vs. 5) of the ITC-2019 benchmark than the basic algorithms introduced in [1] within the same computational budget. Being only 4 hours, this budget is small compared to the related works (see Section 3) and yet goes a long way on the ITC-2019 instances. Additionally, even late in 12 hour runs, the algorithm keeps improving, albeit slowing down. The Luby Restart Strategy [18] in the HRSHCL causes this observed performance improvement. Besides the experimental evidence, we also provide the theoretical underpinning of why the restarts lead to better results.

The ITC-2019 remains highly interesting and challenging due to its many real-world constraints. We will continue our research utilizing other metaheuristic approaches. In our future work, we will explore methods to adaptively configure the run-length parameter ξ of HRSHCL. Furthermore, we will explore softer methods to escape from local optima. Instead of restarts, we will apply Simulated Annealing [14,29], which sometimes accepts worsening moves during the search. Finally, we will compare search operators that span a larger neighborhood, applying smaller changes with great and larger changes with low probability.

References

1. Abdipoor, S., Li, M., Goh, S.L., Yaakob, R., Hao, L., Abdullah, S.: Distance to feasibility: Performance assessment of baseline metaheuristics on university course timetabling with student sectioning. In: IEEE 13th Conference on Systems, Process & Control (ICSPC'2025). pp. 49–54. IEEE (2025)
2. Abdipoor, S., Yaakob, R., Goh, S.L., Abdullah, S.: Meta-heuristic approaches for the university course timetabling problem. Intelligent Systems with Applications **19**, 200253 (2023)
3. Abdipoor, S., Yaakob, R., Goh, S.L., Abdullah, S., Hamdan, H., Kasmiran, K.A.: Optimality vs. generality: Performance assessment of meta-heuristics in educational timetabling. IEEE Access **13**, 75679–75696 (Apr 2025)
4. Burke, E.K., Burke, E.K., Kendall, G., Kendall, G.: Search methodologies: introductory tutorials in optimization and decision support techniques. Springer (2014)
5. Ceschia, S., Di Gaspero, L., Schaerf, A.: Educational timetabling: Problems, benchmarks, and state-of-the-art results. European Journal of Operational Research **308**(1), 1–18 (2023)
6. Chen, M.C., Goh, S.L., Sabar, N.R., Kendall, G., et al.: A survey of university course timetabling problem: perspectives, trends and opportunities. IEEE Access **9**, 106515–106529 (2021)
7. Dalkılıç, Ş.B., Özgür, A., Erdem, H.: Balancing exploration and exploitation in genetic algorithm optimization: a novel selection operator: Şb dalkılıç et al. Evolutionary Intelligence **18**(2), 40 (2025)
8. Das, A., Rai, R., Sasmal, B., Dhal, K.G., Khurma, R.A., Saha, R.: Metaheuristic algorithms since 2020: Development, taxonomy, analysis, and applications. Archives of Computational Methods in Engineering pp. 1–69 (2025)
9. Davison, M., Kheiri, A., Zografos, K.G.: Modelling and solving the university course timetabling problem with hybrid teaching considerations. Journal of Scheduling **28**(2), 195–215 (2025)
10. Er-rhaimini, K.: Forest growth optimization for solving timetabling problems. Proceedings of the International Timetabling Competition (2019)
11. Goh, S.L., Kendall, G., Sabar, N.R.: Improved local search approaches to solve the post enrolment course timetabling problem. European Journal of Operational Research **261**(1), 17–29 (2017)
12. Gu, X., Krish, M., Sohail, S., Thakur, S., Sabrina, F., Fan, Z.: From integer programming to machine learning: A technical review on solving university timetabling problems. Computation **13**(1), 10 (2025)
13. Holm, D.S., Mikkelsen, R.Ø., Sørensen, M., Stidsen, T.J.: A graph-based mip formulation of the international timetabling competition 2019. Journal of Scheduling **25**(4), 405–428 (2022)
14. Kirkpatrick, S., Gelatt, Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Science Magazine **220**(4598), 671–680 (1983)
15. Lemos, A., Monteiro, P.T., Lynce, I.: Introducing UniCorT: An iterative university course timetabling tool with MaxSAT. Journal of Scheduling **25**(4), 371–390 (2022)
16. Lewis, R., Thompson, J.: Analysing the effects of solution space connectivity with an effective metaheuristic for the course timetabling problem. European Journal of Operational Research **240**(3), 637–648 (2015)
17. Lorenz, J.H.: Restart strategies in a continuous setting. Theory of Computing Systems **65**, 1143–1164 (2021)

18. Luby, M., Sinclair, A., Zuckerman, D.: Optimal speedup of Las Vegas algorithms. Information Processing Letters **47**, 173–180 (Sep 1993)
19. Mikkelsen, R.Ø., Holm, D.S.: A parallelized matheuristic for the international timetabling competition 2019. Journal of Scheduling **25**(4), 429–452 (2022)
20. Müller, T., Rudová, H., Müllerová, Z.: Real-world university course timetabling at the international timetabling competition 2019. Journal of Scheduling **28**(2), 247–267 (2025)
21. Müller, T., Rudová, H., Müllerová, Z., et al.: University course timetabling and international timetabling competition 2019. In: 12th international conference on the practice and theory of automated timetabling (PATAT-2018). vol. 1, pp. 5–31 (2018)
22. Pokutta, S.: Restarting algorithms: Sometimes there is free lunch. In: Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR'2020). pp. 22–38. Springer, Cham (2020)
23. Rajwar, K., Deep, K., Das, S.: An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges. Artificial Intelligence Review **56**(11), 13187–13257 (2023)
24. Rappos, E., Thiémard, E., Robert, S., Hêche, J.F.: A mixed-integer programming approach for solving university course timetabling problems. Journal of Scheduling **25**(4), 391–404 (2022)
25. Shaikh, M.S., Raj, S., Zheng, G., Xie, S., Wang, C., Dong, X., Lin, Y., Wang, C., Junejo, N.U.R.: Applications, classifications, and challenges: A comprehensive evaluation of recently developed metaheuristics for search and analysis. Artificial Intelligence Review **58**(12), 1–110 (2025)
26. Steiner, E., Pferschy, U., Schaerf, A.: Curriculum-based university course timetabling considering individual course of studies. Central European Journal of Operations Research **33**(1), 277–314 (2025)
27. Stützle, T., Ruiz, R.: Iterated local search. In: Handbook of heuristics, pp. 779–807. Springer (2025)
28. Sylejmani, K., Gashi, E., Ymeri, A.: Simulated annealing with penalization for university course timetabling. Journal of Scheduling **26**(5), 497–517 (2023)
29. Weise, T.: An Introduction to Optimization Algorithms. free online book (Dec 2020), https://thomasweise.github.io/aitoa
30. Weise, T., Wu, Y., Chiong, R., Tang, K., Lässig, J.: Global versus local search: The impact of population sizes on evolutionary algorithm performance. Journal of Global Optimization **66**(3), 511–534 (Nov 2016)
31. Weise, T., Wu, Z., Li, X., Chen, Y.: Frequency fitness assignment: Making optimization algorithms invariant under bijective transformations of the objective function value. IEEE Transactions on Evolutionary Computation **25**(2), 307–319 (2021)
32. Weise, T., Wu, Z., Wagner, M.: An improved generic bet-and-run strategy with performance prediction for stochastic local search. In: 33rd AAAI Conference on Artificial Intelligence (AAAI'2019). pp. 2395–2402. AAAI Press, Palo Alto, CA, USA (2019)
33. Xiang, K., Hu, X., Yu, M., Wang, X.: Exact and heuristic methods for a university course scheduling problem. Expert Systems with Applications **248**, 123383 (2024)

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

A Genetic Algorithm with Surrogate-Assisted Dynamic Programming for Maintenance Constrained Parallel Machine Scheduling

Fatima Iken[1 [0009−0001−7809−7485]], Abdennour Azerine[1 [0000−0003−2460−3140]], Mahmoud Golabi[1 [0000−0002−3314−2051]], and Lhassane Idoumghar[1 [0000−0001−8853−3968]]

Université de Haute-Alsace, IRIMAS UR 7499, F-68100 Mulhouse, France
{firstname.lastname}@uha.fr

Abstract. This paper addresses the unrelated parallel-machine scheduling problem with non-resumable maintenance under a makespan minimization objective. Maintenance periods restrict machine availability and create a strong coupling between job assignment and sequencing decisions, making solution evaluation computationally expensive. We propose a decomposition-based hybrid genetic algorithm in which the evolutionary search determines the assignment of jobs to machines, thereby defining for each machine the subset of jobs it must process. Given this assignment, each machine-level subproblem is solved optimally using a pseudo-polynomial dynamic programming procedure formulated as a 0–1 knapsack problem. While this exact evaluation ensures accuracy, it constitutes the main computational bottleneck. To reduce this cost, we introduce a surrogate-assisted evaluation mechanism that approximates makespan values and selectively replaces exact computations during the search. The surrogate models are trained online and updated to reflect the evolving search region. Computational experiments on benchmark instances show that the proposed approach achieves substantial runtime reductions while preserving high solution quality. These results demonstrate the effectiveness of combining exact decomposition with data-driven approximation for maintenance-constrained unrelated parallel-machine scheduling.

Keywords: Unrelated parallel machine scheduling · maintenance constraints · genetic algorithm · dynamic programming · surrogate models.

1 Introduction

Scheduling allocates jobs to limited processing resources over time under operational constraints, with the objective of optimizing one or more performance criteria [16]. In manufacturing and production systems, this problem frequently arises in parallel machine environments, where multiple machines operate concurrently, and jobs must be distributed among them [4]. Depending on machine characteristics, such environments are typically classified as identical, uniform,

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

or unrelated machines, with the latter representing the most general case in which processing times depend on both the job and the selected machine [16]. In these settings, minimizing the makespan—the completion time of the last job—remains one of the most widely studied objectives due to its direct impact on throughput and resource utilization [16].

Classical scheduling models generally assume continuous machine availability. In practice, however, machines are subject to preventive maintenance periods that interrupt production [12]. When maintenance is non-resumable, jobs must be processed entirely before or after the unavailability [22]. This constraint partitions the time horizon into disjoint availability intervals and fundamentally alters the structure of the scheduling problem. For a given machine, the decision of which jobs can be completed before maintenance becomes a capacity-constrained selection problem, introducing a knapsack-type structure and eliminating the order-independence property commonly exploited in makespan scheduling [16].

The resulting problem combines two distinct sources of computational complexity. On the one hand, the unrelated parallel machine scheduling problem $Rm||C_{\max}$ is strongly NP-hard. On the other hand, the presence of non-resumable maintenance introduces additional combinatorial structure at the machine level: even in the single-machine case, the problem reduces to the Subset Sum problem and is therefore weakly NP-hard [11]. These results highlight that assignment decisions and maintenance-induced scheduling constraints contribute differently to the overall complexity and must be addressed jointly.

Despite this challenge, most studies on maintenance-constrained scheduling focus on single-machine environments [12]. Research on parallel machines has mostly addressed identical or uniform cases, while the unrelated-machine setting remains underexplored. Kurt and Çetinkaya [10] studied unrelated parallel machines with non-resumable maintenance using a mixed-integer formulation and heuristic. However, exact formulations are known to struggle with larger instances, limiting their applicability to large-scale problems [15].

The problem naturally decomposes into two layers. The first handles job-to-machine assignment, distributing workload across heterogeneous resources. The second addresses per-machine scheduling of assigned jobs within the availability windows imposed by maintenance. For a fixed assignment, the machine-level problem reduces to selecting a subset of jobs that can be processed before maintenance under a capacity constraint, which can be solved optimally using dynamic programming [19]. However, while dynamic programming provides exact solutions for individual machines, it cannot handle the combinatorial explosion induced by assignment decisions across multiple machines.

This observation motivates hybrid approaches that combine global search over assignment decisions with exact resolution of machine-level subproblems. Metaheuristics, and in particular genetic algorithms, have proven effective for exploring large assignment spaces in parallel machine scheduling [1, 21]. Nevertheless, when exact evaluation procedures are embedded within such frameworks, repeated fitness computation becomes the dominant computational cost. To address this issue, surrogate-assisted optimization has been proposed, where

machine learning models approximate expensive evaluation functions and reduce the number of exact computations required [8]. Despite their potential, such approaches remain largely unexplored for maintenance-constrained scheduling on unrelated parallel machines.

We study the unrelated parallel machine scheduling problem with a single non-resumable maintenance interval per machine under makespan minimization. A decomposition-based hybrid framework uses a genetic algorithm (GA) to assign jobs to machines, with each machine-level subproblem solved optimally via a knapsack-based dynamic programming procedure. To reduce repeated exact evaluations, a surrogate-assisted mechanism approximates makespan values during the search, balancing solution quality and computational efficiency while addressing scalability limits of existing methods.

The remainder of the paper is organized as follows. Section 2 defines the problem, Section 3 presents the solution methodology, and Section 4 details the exact evaluation procedure. Section 5 introduces the surrogate-assisted framework, while Sections 6 and 7 present and discuss the computational results. Section 8 concludes the paper.

2 Problem Definition

We consider n independent jobs $\mathcal{J} = \{J_1, \ldots, J_n\}$ to be scheduled on m unrelated parallel machines $\mathcal{M} = \{M_1, \ldots, M_m\}$. Each job J_j has a processing time p_{ij} on machine M_i, where $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$. All jobs are available at time zero, must be assigned to exactly one machine, and preemption is not allowed.

Each machine M_i is subject to a single maintenance period $[\tau_{1i}, \tau_{2i}]$, during which it is unavailable for processing. Jobs are non-resumable: any job that cannot complete before τ_{1i} must start no earlier than τ_{2i}. Let $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ denote an assignment vector, where $\sigma_j \in \{1, \ldots, m\}$ indicates the machine to which job J_j is assigned. For a given assignment σ, let $J_i(\sigma) = \{J_j : \sigma_j = i\}$ denote the set of jobs assigned to machine M_i. The makespan on a specific machine M_i is denoted by $C_i^{\max}(\sigma)$. The objective is to determine a feasible assignment σ that minimizes the global makespan, denoted as: $C_{\max}(\sigma) = \max_{i \in \{1,\ldots,m\}} \{C_i^{\max}(\sigma)\}$. Using the standard three-field scheduling notation [6], the problem is denoted by $Rm \mid nr\text{-}maint \mid C_{\max}$.

3 Solution Methodology

We employ a genetic algorithm [7] to guide the search for the job-to-machine assignment, while all maintenance-related considerations are handled within the fitness evaluation.

3.1 Solution encoding

In the genetic algorithm, each solution is encoded as a vector $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$, where each gene $\sigma_j \in \{1, \ldots, m\}$ specifies the machine assigned to job J_j, pro-

viding a direct integer encoding tailored to the $R \mid nr\text{-}maint \mid C_{\max}$ problem. As each job is assigned to exactly one machine, any chromosome σ defines a feasible assignment, with the decoding step guaranteeing feasibility under the non-resumable constraint by scheduling jobs sequentially and delaying them when necessary to start only after the maintenance window.

For each machine M_i and chromosome σ, the associated job set is $J_i(\sigma) = \{J_j \in J \mid \sigma_j = i\}$, where $i = 1, \ldots, m$. Each set $J_i(\sigma)$ is processed in a fixed order, with completion times computed under the maintenance interval $[\tau_{1i}, \tau_{2i}]$ of machine M_i. Whenever a job overlaps the start of maintenance, its processing is delayed until time τ_{2i}, consistent with the non-resumable maintenance constraint. This encoding guarantees feasibility by construction, eliminating repair mechanisms, while the assignment–scheduling separation enables independent per-machine fitness evaluation under the maintenance constraints.

3.2 Algorithm Overview

The GA follows a standard generational scheme with elitism, with parameters determined through preliminary experimentation, comparing tournament and roulette-wheel selection, one-point and uniform crossover, and swap and uniform mutation operators.

According to this configuration, the process begins with an initial population of 70 candidate solutions, each generated by sampling its gene σ_j uniformly from $\{1, \ldots, m\}$ to ensure diversity. For each individual σ, fitness is evaluated by decomposing the problem into m independent single-machine subproblems. For each machine M_i, the algorithm computes a completion time $C^i_{\max}(\sigma)$ within its maintenance interval $[\tau_{1i}, \tau_{2i}]$, and the global makespan is defined as $C_{\max}(\sigma) = \max_i C^i_{\max}(\sigma)$.

Parents are selected using tournament selection and recombined through uniform crossover with probability $p_c = 0.75$ to generate new offspring solutions. Diversity is maintained using swap mutation with probability $p_m = 0.10$ [2]. At each generation, the best individual is preserved through elitism, and its objective value also serves as a reference value in the surrogate-assisted evaluation mechanism described in Section 4.

4 Single-Machine Evaluation

This section formulates the single-machine subproblem for a fixed assignment σ, establishes its knapsack reduction, and presents the exact dynamic programming algorithm.

4.1 Fixed-Assignment Machine Subproblems

For a given machine M_i, let $J_i(\sigma)$ denote the set of assigned jobs, and let $P_i = \{p_{ij} : J_j \in J_i(\sigma)\}$ be the multiset of processing times. The scheduling problem on M_i reduces to deciding which jobs to execute before and after its maintenance interval $[\tau_{1i}, \tau_{2i}]$.

4.2 Optimal Subset Selection via Knapsack Reduction

Proposition 1. *Given a fixed assignment σ and machine M_i, minimizing $C^i_{\max}(\sigma)$ is equivalent to selecting a subset $B_i \subseteq J_i(\sigma)$ of jobs to maximize the total processing time completed before maintenance, subject to the capacity constraint:*

$$\max_{B_i \subseteq J_i(\sigma)} \sum_{J_j \in B_i} p_{ij} \quad s.t. \quad \sum_{J_j \in B_i} p_{ij} \leq \tau_{1i}. \tag{1}$$

Proof. Let $k_i = |J_i(\sigma)|$. Consider machine M_i with its assigned jobs $J_i(\sigma)$ and corresponding processing times P_i. Since all jobs are available at time zero and no setup times are considered, scheduling reduces to deciding which subset of jobs completes before maintenance.

If the total load satisfies $\sum_{p \in P_i} p \leq \tau_{1i}$, all jobs can be processed before maintenance, giving $C^i_{\max}(\sigma) = \sum_{p \in P_i} p$, which is trivially optimal.

Otherwise, let $B_i \subseteq J_i(\sigma)$ denote the subset of jobs completed before τ_{1i}, and the remaining jobs start after τ_{2i}. Since jobs are non-preemptive and available from time zero, they can be scheduled consecutively in any order within these intervals without increasing the completion time. Thus, the completion time is: $C^i_{\max}(\sigma) = \tau_{2i} + \sum_{J_j \in J_i(\sigma) \setminus B_i} p_{ij}$.

Because the total load is fixed, minimizing $C^i_{\max}(\sigma)$ is equivalent to maximizing the total processing time scheduled before maintenance, which is exactly the knapsack problem in (1).

The induced single-machine subproblem is weakly NP-hard [13], as Proposition 1 establishes its equivalence to a 0–1 knapsack problem [14].

4.3 Optimal Evaluation via Dynamic Programming

The exact optimum of (1) is computed via a pseudo-polynomial dynamic programming method with $O(k_i \tau_{1i})$ time and memory complexity [9], summarized in Algorithm 1, returning the best packed value and, when required, the corresponding subset via backtracking.

When τ_{1i} is large, this exact evaluation may be expensive in both time and memory, motivating the surrogate-assisted strategy described in Section 5.

Exact Subproblem Evaluation: Define $L_i(\sigma) = \sum_{J_j \in J_i(\sigma)} p_{ij}$ as the total load on machine M_i, and let $V_i(\sigma)$ denote the optimal value of (1). Consequently,

$$C^i_{\max}(\sigma) = \begin{cases} L_i(\sigma), & \text{if } L_i(\sigma) \leq \tau_{1i}, \\ \tau_{2i} + L_i(\sigma) - V_i(\sigma), & \text{otherwise,} \end{cases}$$

and the overall makespan is given by $C_{\max}(\sigma) = \max_i \{C^i_{\max}(\sigma)\}$.

5 Surrogate-Assisted Evaluation

This section presents the surrogate-assisted mechanism used to approximate makespan values and reduce repeated exact evaluation costs.

5.1 Surrogate Model and Training Strategy

A surrogate-assisted strategy is employed to approximate the mapping $f : \sigma \mapsto C_{\max}(\sigma)$, where $\sigma = (\sigma_1, \ldots, \sigma_n) \in \{1, \ldots, m\}^n$ denotes a job-to-machine assignment vector and $C_{\max}(\sigma) \in \mathbb{R}_+$ is the exact makespan obtained through the evaluation procedure described in Section 4. The surrogate is trained online during the genetic algorithm using pairs (σ, y), where σ represents the raw integer encoding of an assignment and $y = C_{\max}(\sigma)$ is the corresponding exact makespan.

The integer encoding is used directly as input features to maintain consistency with the genetic representation. To control training cost and mitigate concept drift as the search progresses, the training dataset is maintained as a sliding window containing at most $N_{\max} = 2000$ of the most recent exact evaluations, with older samples discarded. This online data management strategy preserves model adaptability to the current search region while maintaining bounded computational overhead.

5.2 Evaluation Policy and Safeguards

The surrogate becomes operational after a three-generation initialization period E_0, during which all candidates are exactly evaluated to build the foundational training dataset. After activation, the model predicts makespan values $\tilde{C}_{\max}(\sigma)$ for most candidates and is periodically retrained at ten-generation intervals to reflect the evolving search trajectory.

Exact evaluation is triggered by a dual criterion: any solution whose surrogate estimate falls below $C_{\max}^{\text{best}}$ receives mandatory exact evaluation. In addition, a random exact evaluation is triggered with probability $\rho = 0.1$ to enforce exact evaluation, preserve exploration, and mitigate systematic prediction errors. Each exact evaluation contributes $(\sigma, C_{\max}(\sigma))$ to the training repository, updating the best-known solution when improvements occur, while remaining candidates use the surrogate prediction as fitness.

Algorithm 1 Exact makespan evaluation for a fixed assignment σ

Require: Assignment vector σ, processing times (p_{ij}), maintenance (τ_{1i}, τ_{2i}) for all machines

for $i \leftarrow 1$ to m **do**
  $\mathcal{P}_i \leftarrow \{p_{ij} : \sigma_j = i\}$
  $L_i \leftarrow \sum_{p \in \mathcal{P}_i} p$
  if $L_i \leq \tau_{1i}$ **then**
    $C_{\max}^i \leftarrow L_i$
  else
    $V_i \leftarrow \textsc{KnapsackDP}(\mathcal{P}_i, \tau_{1i})$
    $C_{\max}^i \leftarrow \tau_{2i} + (L_i - V_i)$
  end if
end for
return $\max_i \{C_{\max}^i\}$

5.3 Surrogate Regression Models

Five surrogate models are evaluated: Random Forests (RF) [3] and Gradient Boosting (GB) [5] for piecewise interactions; Gaussian Processes (GP) [18] for uncertainty quantification; Support Vector Regression (SVR) [20] for kernel-based regression; and Multi-Layer Perceptrons (MLP) for flexible nonlinear approximation. For each model, the hyperparameters are set to default values, as commonly adopted in the literature [17]. Results are reported in Section 6.

6 Computational Experiments

This section assesses the performance of the proposed GA under both exact and surrogate-assisted fitness evaluation. We compare a baseline genetic algorithm using exact dynamic programming with the surrogate-assisted variant introduced in Section 5, across five regression models, on a set of synthetically generated instances. Experiments were conducted on a workstation equipped with an Intel Xeon W-2265 CPU @ 3.50 GHz, 64 GB RAM, running Windows 11 Pro and Python 3.13.

6.1 Instance generation and maintenance setup

Benchmark instances are defined by a pair (m, n), where m is the number of machines and n the number of jobs, across seven configurations: $(m, n) \in \{(2, 20), (2, 50), (3, 50), (2, 70), (3, 70), (2, 100), (3, 100)\}$. Processing times p_{ij} are sampled independently for each machine-job pair, with 20 instances per configuration.

For each machine M_i, the total hypothetical load $L_i = \sum_{j=1}^{n} p_{ij}$ is computed. The maintenance start time is defined as $\tau_{1i} = \lfloor L_i \alpha \rfloor$, where $\alpha \sim U(0.3, 0.7)$, positioning maintenance within the central portion of the processing horizon. The maintenance duration is sampled uniformly from $[d_{\min}, d_{\max}]$, where $d_{\min} = \max\{p_{\min}, \lfloor 0.15\, L_i \rfloor\}$ and $d_{\max} = \max\{d_{\min} + 1, \lfloor 0.40\, L_i \rfloor\}$, and $\tau_{2i} = \tau_{1i} + \text{duration}$. Scaling parameters with L_i preserve comparable disruption levels across instance sizes while naturally yielding heterogeneous maintenance windows.

Performance comparison Across all configurations reported in Table 1, all surrogate-assisted variants achieve substantial runtime reductions relative to the baseline GA, with GA+GB and GA+GP consistently emerging as the most balanced surrogates. At $n = 20$, GA+GB reduces runtime from 435.70 to 85.40 seconds with a makespan increase of only 20.29 units, while GA+GP achieves a runtime of 82.85 seconds with a makespan of 1910.74. This advantage scales with problem size; at $n = 100$ and $m = 2$, GA+GB and GA+GP reduce runtime to 355.72 and 348.75 seconds, respectively, compared to 2594.10 seconds for the baseline GA.

Table 1: Mean results by configuration. Runtime is reported in seconds.

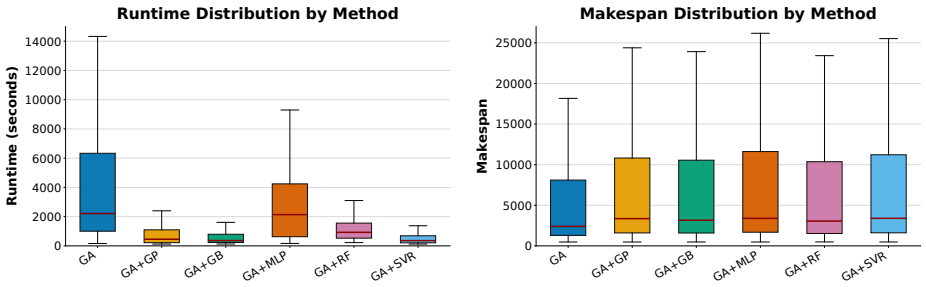
n	m	Method	Makespan	Runtime	Epochs
20	2	GA	1850.21	435.70	124.3
		GA+GP	1910.74	82.85	161.7
		GA+GB	1870.50	85.40	175.4
		GA+MLP	1927.74	1262.56	144.1
		GA+RF	1922.12	297.63	164.6
		GA+SVR	1933.29	73.37	159.5
50	2	GA	4076.21	3164.36	171.3
		GA+GP	4378.81	709.21	316.7
		GA+GB	4554.22	470.72	203.8
		GA+MLP	4959.80	4035.49	190.9
		GA+RF	4560.32	854.18	238.3
		GA+SVR	5068.11	465.21	213.1
50	3	GA	2341.36	4321.69	241.3
		GA+GP	2772.82	1062.15	430.1
		GA+GB	2763.69	592.76	232.8
		GA+MLP	3053.56	3354.68	164.3
		GA+RF	2663.18	1357.50	372.9
		GA+SVR	2923.33	636.63	247.1
70	2	GA	6958.28	7984.61	207.1
		GA+GP	7608.22	1974.78	382.2
		GA+GB	7539.94	1047.22	209.7
		GA+MLP	8580.25	5514.51	164.4
		GA+RF	7448.25	1888.72	315.8
		GA+SVR	8418.94	1197.02	268.2
70	3	GA	3043.72	10729.39	276.0
		GA+GP	4581.17	1367.62	378.0
		GA+GB	4403.97	966.15	182.8
		GA+MLP	5100.56	4522.27	145.4
		GA+RF	4159.68	2281.25	400.4
		GA+SVR	4595.97	1409.12	276.9
100	2	GA	12432.42	2594.10	78.2
		GA+GP	15611.40	348.75	65.5
		GA+GB	14039.77	355.72	75.1
		GA+MLP	16020.90	484.98	41.9
		GA+RF	14412.43	735.57	89.6
		GA+SVR	15943.81	234.44	55.0
100	3	GA	4212.29	9166.66	209.3
		GA+GP	6602.39	1291.16	216.1
		GA+GB	6144.85	848.69	118.0
		GA+MLP	6575.95	2986.03	87.7
		GA+RF	5975.68	1864.21	210.6
		GA+SVR	6570.04	1053.83	130.1

7 Results and Discussion

This section compares the baseline GA against five surrogate-assisted variants. The analysis covers computational efficiency, solution quality, and surrogate prediction accuracy across all tested configurations.

As shown in Figure 1, replacing exact dynamic programming with a surrogate model consistently reduces runtime across all five models, bringing worst-case GA runtimes, over 6000 s, down to a markedly lower and more stable range, with GA+GB and GA+GP performing best (Figure 1a), at the cost of only a modest increase in $C_{\max}$, where GA+RF and GA+GP remain closest to the baseline (Figure 1b).

Surrogate Accuracy Across Problem Scales: The surrogate models remain highly accurate across all problem sizes, with the best reaching mean accuracies near 0.9 in some instances, as shown in Figure 2. Among them, GA+GP, GA+RF, and GA+GB consistently deliver the highest accuracy across all job counts.



(a) Runtime Distribution by Method (b) Makespan Distribution by Method

Fig. 1: Runtime and Makespan Distribution Across Surrogate Methods.

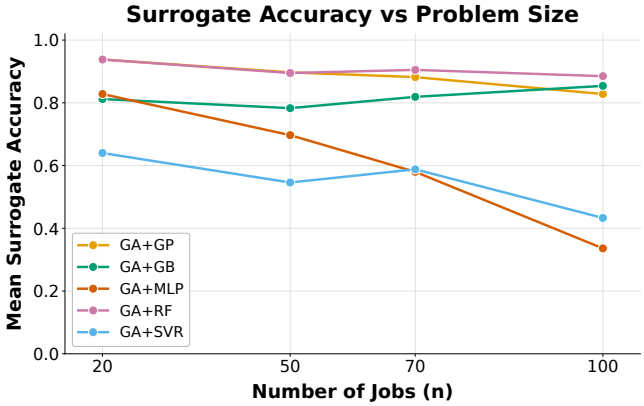
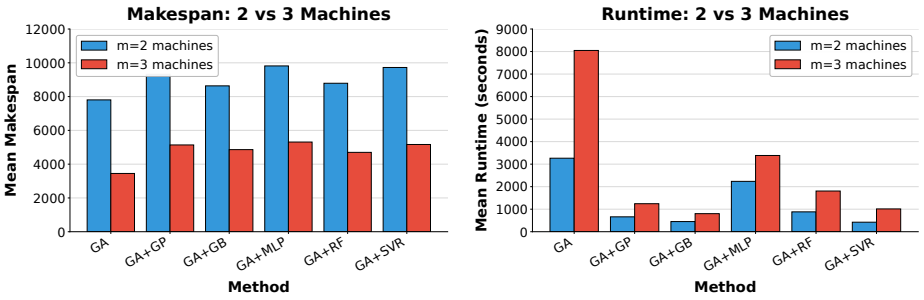


Fig. 2: Surrogate Accuracy.

Effect of Problem Scale: Machine Count Increasing machines from $m = 2$ to $m = 3$, improves $C_{\max}$ across all methods while surrogate-assisted variants remain computationally efficient, as shown in Figure 3. GA+GB stays well below 1000 seconds for both configurations, while the baseline GA climbs to nearly 8000 seconds, underlining the practical value of surrogate assistance as the assignment space expands.

Impact on makespan and solution quality: For small to medium instances n = 20 and n = 50, all surrogate methods yield values of the makespan $C_{\max}$ that are almost indistinguishable from those of the baseline GA, so the quality–efficiency trade-off remains highly favorable; beyond (n = 50), the methods start to diverge, with GA+RF and GA+GP staying closest to the baseline and GA+MLP showing the most noticeable degradation as shown in Figure 4.



(a) Effect of machine count on makespan (b) Effect of machine count on runtime

Fig. 3: Effect of Machine Count on Makespan and Runtime

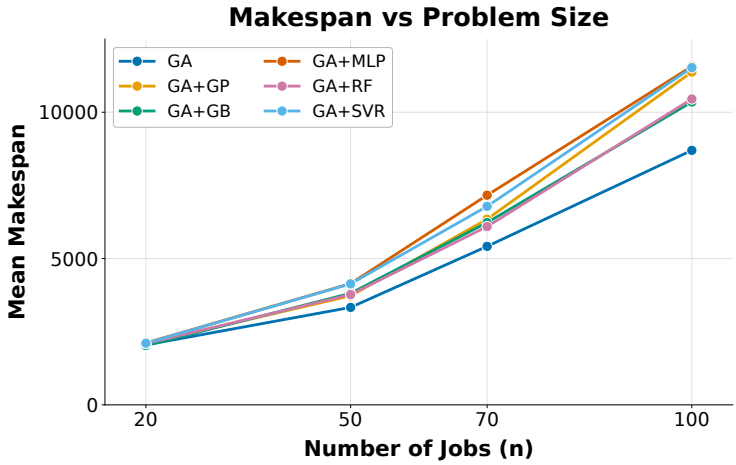


Fig. 4: Effect of Problem Size on Solution Quality Across Surrogate Methods.

8 Conclusion and future perspectives

This paper addressed the unrelated parallel-machine scheduling problem with one non-resumable maintenance window per machine under makespan minimization. To tackle the high computational cost of exact fitness evaluation, we proposed a hybrid framework combining a genetic algorithm for job-to-machine assignment with exact dynamic programming for machine-level subproblem resolution, augmented by a surrogate-assisted evaluation framework benchmarking five regression models: Gaussian Process, Gradient Boosting, Multilayer Perceptron, Random Forest, and Support Vector Regression. The experiments show that all surrogate-assisted variants substantially reduce runtime compared to the baseline GA, with GA+GB and GA+GP consistently emerging as the most balanced in terms of solution quality and computational efficiency. GA+RF and GA+GP maintain the highest prediction accuracy across all problem sizes,

while GA+MLP and GA+SVR degrade at larger scales. Adding a third machine consistently improves $C_{\max}$ across all methods, and surrogate-assisted variants, particularly GA+GB, handle the added complexity with far smaller runtime increases than the baseline GA.

This work opens several promising directions for future research. Adaptive surrogate strategies that dynamically balance exact and approximate evaluations based on search progress could enhance solution quality without sacrificing computational efficiency. Extending the framework to multiple or resumable maintenance periods would introduce richer problem structures and provide a more demanding test for surrogate models. Incorporating local search operators, such as job reassignment following surrogate-guided exploration, could further refine schedule quality. Finally, applying the framework to identical and uniform parallel machine settings would enable assessment of its generalizability across a broader class of scheduling problems.

References

1. Adan, J.: A hybrid genetic algorithm for parallel machine scheduling with setup times: A comparative study of metaheuristics on large problem instances. Journal of Intelligent Manufacturing **33**(7), 2059–2073 (2022)
2. Azerine, A., Golabi, M., Oulamara, A., Idoumghar, L.: Enhancing electric vehicle charging schedules: A surrogate-assisted approach. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. pp. 183–186 (2024)
3. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
4. Cheng, T., Sin, C.: A state-of-the-art review of parallel-machine scheduling research. European Journal of Operational Research **47**(3), 271–292 (1990)
5. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of Statistics **29**(5), 1189–1232 (2001)
6. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics **5**, 287–326 (1979)
7. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
8. Ikonen, T.J., Heljanko, K., Harjunkoski, I.: Surrogate-based optimization of a periodic rescheduling algorithm. AIChE Journal **68**(6), e17656 (2022)
9. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, Berlin (2004). https://doi.org/10.1007/978-3-540-24777-7
10. Kurt, A., Çetinkaya, F.C.: Unrelated parallel machine scheduling under machine availability and eligibility constraints to minimize the makespan of non-resumable jobs. International Journal of Industrial Engineering and Management **15**(1), 18–33 (2024)
11. Lee, C.Y.: Machine scheduling with an availability constraint. Journal of global optimization **9**(3), 395–416 (1996)
12. Low, C., Ji, M., Hsu, C.J., Su, C.T.: Minimizing the makespan in a single machine scheduling problems with flexible and periodic maintenance. Applied Mathematical Modelling **34**(2), 334–342 (2010)
13. Luo, W., Chen, L., Zhang, G.: Approximation algorithms for scheduling with a variable machine maintenance. In: International Conference on Algorithmic Applications in Management. pp. 209–219. Springer (2010)

14. Luo, W., Liu, F.: On single-machine scheduling with workload-dependent maintenance duration. Omega **68**, 119–122 (2017)
15. Pacheco, J., Ángel-Bello, F., Álvarez, A.: A multi-start tabu search method for a single-machine scheduling problem with periodic maintenance and sequence-dependent set-up times. Journal of Scheduling **16**(6), 661–673 (2013)
16. Pinedo, M.L.: Scheduling, vol. 29. Springer (2012)
17. Prieto, N.V., Garrido-Merchán, E.C.: Default machine learning hyperparameters do not provide informative initialization for bayesian optimization. arXiv preprint arXiv:2602.08774 (2026)
18. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press (2006)
19. Sadfi, C., Penz, B., Rapine, C.: A dynamic programming algorithm for the single machine total completion time scheduling problem with availability constraints. In: Eighth international workshop on project management and scheduling (2002)
20. Smola, A.J., Schölkopf, B.: A tutorial on support vector regression. Statistics and computing **14**(3), 199–222 (2004)
21. Vallada, E., Ruiz, R.: A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. European Journal of Operational Research **211**(3), 612–622 (2011)
22. Zou, J., Yuan, J.: Single-machine scheduling with maintenance activities and rejection. Discrete Optimization **38**, 100609 (2020)

Synthetic Instance Generation for the Radiotherapy Scheduling Problem

Chiara Camilla Rambaldi Migliore[1,2] [0009−0000−1581−6086], Nysret Musliu[3] [0000−0002−3992−8637], Giovanni Iacca[1] [0000−0001−9723−1830], and Marco Roveri[1] [0000−0001−9483−3940]

[1] University of Trento, Italy
[2] University of Pisa, Italy
{cc.rambaldimigliore,marco.roveri,giovanni.iacca}@unitn.it
[3] Technische Universität Wien, Austria
nysret.musliu@tuwien.ac.at

Abstract. The Radiotherapy Scheduling Problem (RTSP) concerns finding an optimal appointment schedule for patients undergoing radiation treatments. Given its considerable impact on clinical outcomes, this problem has been studied in previous research, relying on datasets obtained from various hospital settings; however, such real-world datasets are rarely made public–mainly for ethical approval issues–while synthetic problem instances are rare and not standardised, which makes the comparison between scheduling algorithms harder. To address this gap, we propose a Benchmark Instance Management Tool that comprehends 1) a generator of new instances that samples from statistical distributions tuned on two real-world reference datasets from a big Belgian cancer centre and a small Italian radiotherapy unit, 2) a set of solvers, such as Mixed Integer Linear Programming, Simulated Annealing and two heuristics, and 3) a validator to check solution feasibility. Moreover, we propose a standardised input and output format. The generated instances are compared to real-world data by means of Instance Space Analysis on features extracted directly from instances. The results reveal two sharply separated clusters for the real-world reference datasets in the 2-dimensional feature space, while the newly generated instances not only populate those clusters but also fill the gap between them.

Keywords: Benchmark · Scheduling · Radiotherapy

1 Introduction

New cancer cases are predicted to exceed 27 million new diagnoses worldwide per year by 2040 [23]. Among the currently available cancer treatments, radiation therapy is widely used and is considered to be effective, especially for some specific types of cancer [15]. It consists of radiation delivery to the tumour areas in a set of fractionated sessions (called *fractions*), and an early start of treatment, followed by a precise planning of sessions, can be crucial for achieving optimal treatment results [9]. Thus, the Radiotherapy Scheduling Problem

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

(RTSP)–which consists of finding optimised appointment schedules balancing patient needs, resource constraints, and treatment efficacy–is a crucial field of study with major social implications.

The RTSP has been addressed by several authors [20] by introducing various problem formulations and algorithms, ranging from exact Operational Research (OR) methods [13, 7, 8] to metaheuristics [4, 22, 21, 2]. However, existing studies usually rely on data collected in specific hospitals, or generated from them, without any specific standard for input, problem formulation, or output. This contingency makes it difficult to compare how different algorithms perform on the RTSP. Moreover, the majority of the datasets are not public–mainly for ethical approval issues–or, when generated, the parameter and sampling distributions are not accurately described, making it difficult to reproduce them. To address this gap, following the state-of-the-art formulation of the problem and the solvers described and used in [7, 14], we present the following contributions:

- We ground our study in two real-world radiotherapy settings: Iridium Netwerk Cancer Centre, a major cancer hub in Antwerp, Belgium [7] (hereafter **Netwerk**), and Azienda Sanitaria Universitaria Integrata del Trentino's radiotherapy unit in Trento, Italy (hereafter **ASUIT**), a smaller but highly active facility. Using these benchmarks, we extract the empirical statistical distributions of patient arrivals, their associated treatment plans, and the occupancy of linear accelerators (LINACs) over a specified time horizon.
- We introduce a new, fully parametrised and customisable instance generator that leverages the above statistical distribution to produce synthetic data closely mirroring real-world cases–bridging the gap between the large cancer-centre dataset from [7] and the smaller ASUIT cohort.
- We embed our generator into a Benchmark Instance Management Tool that not only creates new instances on demand, but also automatically solves them using state-of-the-art solvers derived from [14], validates every solution, and stores all results for seamless analysis and reuse.
- We extract features from **Netwerk**, **ASUIT**, and our newly generated instances, then apply Instance Space Analysis (ISA) [17] to project them into a 2-dimensional feature space. This shows where each instance lies and how effectively the new instances bridge the gap between cases from small and large hospitals.

The ISA results reveal a sharp separation between the **Netwerk** and **ASUIT** datasets, which form distinct clusters in the 2-dimensional feature space, making their difference in facility size clear. The generated instances show that our generator not only fills the gap between these clusters, but also populates the clusters themselves–strong evidence that the statistical analysis successfully captures the key characteristics of the two real-world instances.

The remainder of this article is organised as follows. Section 2 summarises the main related works. Section 3 presents the definition of the RTSP, formalising the problem input, solution, and objective. Section 4 describes the Benchmark Instance Management Tool, starting from the standardisation of the input and output format, continuing with the methodology for building the statistical dis-

tribution from which to sample, and with the description of the parameters the user can provide to the generator, ending with the summary of solvers and the validator. Section 5 introduces the Instance Space Analysis results by comparing the reference datasets with the newly generated ones on a 2-dimensional feature space. Finally, Section 6 draws conclusions and outlines future work.

2 Related Works

Scheduling problems in radiotherapy have been the subject of study for more than two decades. Early works such as [5, 10, 3, 16, 11] addressed variants of the RTSP using data from real cancer centres, but, to the best of our knowledge, none have made their datasets or code publicly available, and the generation processes were absent or only partially described. Throughout the literature, there is a lack of standardised and reproducible benchmarks. One exception is [4], where both a generator and instances were published; unfortunately, the associated website is no longer available. More recent works follow the common approach of generating data from real-world sources, but none of them are standardised, and often, datasets are not even publicly available. [19] provides good detail on the generator's parameters, and [18] subsequently publishes real-world data, albeit in a non-standardised format, while the solver code remains unavailable in both cases. [13] generates instances based on real-world data with explained distributions and publicly available data, but the dataset uses a custom format, and no code is released. Other works [1, 6] share instances derived from a Thai cancer centre, but provide neither the code nor sufficient details on the generation process to allow for replication.

A separate line of work addresses scheduling in ion beam or particle therapy facilities, most notably at MedAustron; [12, 21, 22, 2] all tackle variants of this problem, with some datasets made available. However, the formulation of particle therapy differs in some aspects compared to LINAC-based RTSP (that is, the focus of this work), making it impossible to use those generators and datasets.

The work most closely related to ours is [7], which generates instances from real-world data from the Iridium Netwerk Cancer Centre, in Belgium, and provides both a public real-world-like synthetic dataset and a partially described generation process. However, their work only presents instances from a large facility, and the code is not provided. For this reason, we generate new instances of various sizes and difficulties and provide the scientific community with a Benchmark Instance Management Tool to generate, solve, and validate new instances.

3 Problem definition

The RTSP involves scheduling all patients' **fractions**–i.e., individual radiotherapy sessions delivering a portion of the total prescribed dose–by assigning each a **day**, **time window** (a predefined part of the day), and **machine**, subject to a set of constraints. The formulation and solution quality metric used here follow [7], with a more detailed description provided in [14].

Problem Input. The problem has three main inputs: a list of **patients**, a **time horizon**, and a list of **machines**. Each patient is prescribed with a specific

treatment protocol, which defines the priority, the number of fractions to schedule and their duration, the machines allowed and preferred to perform the treatment, the allowed start days and the target day (i.e., the latest day after which it is considered late to start the treatment). Patients may or may not have a preference for the time window of the day in which to be scheduled. The time horizon is defined by the number of days considered for the scheduling. Each machine has a residual time capacity for each time window of each day of the time horizon, measured in minutes. Some machines may be partially or fully beam-matched, meaning that the same radiotherapy plan can be safely administered by any of them. Machines are considered fully matched when they are located in the same building; if they are in different buildings, they are partially matched (assuming a cancer centre may be located in more than one building).

Solution. A solution to the RTSP consists of the complete appointment calendar for each patient. A feasible solution satisfies the following criteria:

1) **Singularity**: Each patient receives at most one fraction per day, scheduled in exactly one time window of one machine.
2) **Consecutiveness**: Fractions must be scheduled on consecutive weekdays.
3) **Availability**: A fraction can only be assigned to a machine's time window if sufficient capacity remains.
4) **Starting**: Treatment must begin on a protocol-allowed weekday, no earlier than the patient's arrival and no later than $T - n_f$, where T is the time horizon and n_f the total number of the patient's treatment fractions.
5) **Eligibility**: Each patient may only be treated on machines allowed by their protocol.
6) **Protocol precedence**: Among patients sharing the same protocol and same start day, those with an earlier target day must start treatment no later than those with a later one.

Objective. The objective is to minimise a weighted sum of six objectives, plus an offset of 1, as in [7, 14]:

$$\text{Obj. Sum} = 1 + \alpha_1 f_1 + \alpha_2 f_2 + \alpha_3 f_3 + \alpha_4 f_4 + \alpha_5 f_5 + \alpha_6 f_6 \tag{1}$$

where: f_1 is the weighted sum of the days patients are waiting before starting the treatment (the weights are defined by `cost` field defined in Section 4); f_2 is the weighted sum of the days beyond the deadline for patients to start their treatment (the weights are the same as for f_1); f_3 is the number of times patients switch time windows between two fractions; f_4 is the sum of the deviations from the time window preferred by the patients over all fractions; f_5 is the number of patients' fractions scheduled on a non-preferred machine as per the treatment protocol; and, f_6 is the sum of switches between machines that are partially beam-matched, i.e., machines whose treatment plans are interchangeable (they can be thought of as clones), but located in different buildings. The alpha values used for our experiments are $(50, 100, 1, 0, 10, 10)$, corresponding to the first combination discussed in [14].

4 Benchmark Instance Management Tool

4.1 Input format

The format of the standardised dataset output by the generator and expected as input by the solver, is an input parameter file[4]. The input parameter file contains various fields related to *machines* and to *patients*. In the following, we present a description of each of them.

Machines fields. The input file contains simple information fields, such as the `machinesQty` and `timeWindowsQty`, stating the number of machines and the number of time windows for each machine. The machines' maximum capacities `machinesMaxCapacity` and the residual capacities `machinesCapacity` (i.e., the remaining capacity at the start day of the scheduling; the occupied minutes are taken by patients already scheduled on that date) are designed as arrays of length $l = \texttt{machinesQty} \cdot \texttt{timeWindowsQty}$ with index $i = m \cdot \texttt{timeWindowsQty} + tw$ where m and tw represent the machine and the number of time windows targeted. Lastly, a more complex field, `machineBeamMatching`, represents the beam-matching relation between machines: the field is a matrix $\texttt{machinesQty} \times \texttt{machinesQty}$ where each cell (m_i, m_j) can take values in $\{0, 1, 2\}$ if $i \neq j$ and value 2 if $i = j$. The value 0 means that m_i is not beam-matched with m_j, the value 1 means that m_i is *partially* beam-matched with m_j, while the value 2 means that m_i is *completely* beam-matched with m_j.

Patients fields. The `patientsQty` field indicates the total number of patients to be scheduled, while `daysQty` contains the number of days in the time horizon. The `patientsGroupedByProtocol` field is an array of length equal to the number of protocols, where each element is another array filled with the ids of those patients having that same protocol treatment. The patients' ids are ordered ascending by the target day of each patient within the same protocol. This information is then used to ensure the Protocol precedence criterion described in Section 3. A more complex field is `patientInfo`, which–for each patient–is a data structure containing:

- `priority`: integer in $\{1, 2, 3\}$, where 1 denotes the highest clinical urgency.
- `cost`: scalar penalty weight associated with the priority class, set to 10, 3, and 1 for priorities 1, 2, and 3, respectively; it is used to calculate the objective function described in Section 3.
- `dMin`: the earliest day on which the patient can begin the treatment.
- `dTarget`: the clinically recommended latest start day.
- `numFractions`: the total number of treatment fractions specified by patients' protocol.
- `fractionsDuration`: an array of length `numFractions` specifying the duration in minutes of each fraction.
- `allowedDays`: a binary array of length `daysQty` indicating with 1 the days allowed to begin the treatment and with 0 the not allowed ones.

[4] The input parameter file is an ad-hoc data structure stored as a pickle and json file.

– `twPref`: an optional integer that indicates the preferred time window. If the patient does not express a preference, a placeholder "`Null`" value is inserted.

Finally, the field `machineEligibility` is a matrix `patientQty` × `machinesQty` where each cell (p_i, m_j) can take a value in $\{0, 1, 2\}$ where 0 means that machine m_j is not eligible for patient treatment p_i, 1 means that it is allowed, while 2 means it is *preferred* among the ones allowed.

4.2 Solution format

The standardised solution format output by the solvers provided in the tool is a file containing a specific data structure composed by two main fields: `patientAppointments` and `startDays`. The first is a matrix `daysQty` × `patientsQty` where each cell (d_i, p_j) contains a tuple (a, b) where $a = m \cdot$ `timeWindowsQty` $+ tw$ is a combined index indicating the machine and time window in which the patient p_j will receive the treatment fraction, and b is the duration of that fraction. The latter field is an array of length `patientsQty` specifying the start day for each patient, used for efficient calculation of the objective function.

4.3 Reference datasets

The first dataset considered for generator development–i.e., **Netwerk**–is a public one, containing real-world-like synthetic data from the Iridium Netwerk Cancer Centre based in Antwerp, Belgium [7]. It features 10 machines distributed over different buildings and encompasses instances that consider either 2 or 4 time windows. To ground the generator in real-world data from a smaller hospital, we collected, cleaned, and analysed real-world anonymised data (which we refer to as **ASUIT**) we obtained from the Azienda Sanitaria Universitaria Integrata del Trentino radiotherapy unit based in Trento, Italy. This dataset features 4 machines all in the same building and, as **Netwerk**, instances consider both 2 and 4 time windows. To make the **ASUIT** dataset publicly available, we created synthetic, real-world-like instances by sampling from the real-world instances' distribution.

4.4 Generator methodology

The generator tool has been built by analysing the statistical distribution of machines' occupation and patient arrivals of both the **ASUIT** and **Netwerk** reference datasets, in order to have a realistic range for the parameters of the statistical distribution used for the generation.

Machine occupation generation. From the reference datasets, we calculate, for each day d, machine, and time window, the empirical mean and variance of the occupation across similar instances–those with the same machines and time windows quantity–, denoted by $mean_d^i$ and var_d^i, respectively. To capture overdispersion, we use a Negative Binomial($\mathcal{NB}$) distribution to generate the machine occupation for each day, estimating its dispersion parameter via the method of moments. For each day d, we aggregate these estimates across all machines and time windows by taking the median:

$$\hat{r}_d = median\left(r_d^i = \frac{(mean_d^i)^2}{var_d^i - mean_d^i}\right), \quad i \in [0, \text{machine\_qty} \cdot \text{tw\_qty}] \tag{2}$$

The estimator is valid when $var_d^i \neq mean_d^i$; otherwise, the data are not overdispersed and are consistent with a Poisson ($\mathcal{P}$) distribution. In such cases, we store a placeholder instead of r_d^i. Since the mean occupation of all machines' time windows across instances in both reference datasets follows a decreasing pattern over days, we fit a quadratic exponential regression:

$$\mu_d^i = e^{(\beta_0^i + \beta_1^i d + \beta_2^i d^2)}, \quad i \in [0, \text{machine\_qty} \cdot \text{tw\_qty}] \tag{3}$$

where μ_d^i represents the expected occupation on day d for a specific time window of a machine averaged across similar instances, and $\ln(\beta_0)$ controls the initial occupation level. Across all machines' time windows. Only fits with the coefficient of determination $R^2 \geq 0.75$–which is considered enough for a good fitting–are retained. From the accepted fits, we calculate $\hat{\beta}_2 = mean(\beta_2^i)$–which will then be used as is by the generator during the creation process–and we bound β_1 between β_1^{min}, the minimum observed fitted value of β_1, and $\beta_1^{max} = min(0, max(\hat{\beta}_1))$–a range derived from thorough analysis across all machine time windows. The actual values of $\hat{\beta}_0$ and $\hat{\beta}_1$ used by the generator when producing new instances are calculated by means of two parameters given by the user: the initial occupation ω, which determines $\hat{\beta}_0 = \ln(\omega \times \text{maximum\_capacity})$, and the decay velocity of the occupation $\nu \in [0, 1]$, which determines $\hat{\beta}_1 = \nu\beta_1^{\min} + (1 - \nu)\beta_1^{\max}$. The generator, thus, will calculate the estimated daily occupation mean $\hat{\mu}_d$ with the $\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2$ described above, and feed it to the Negative Binomial ($\mathcal{NB}$) if the above calculated $var_d^i \neq mean_d^i$ else to the Poisson ($\mathcal{P}$) to generate the occupation Occ_d for each day–which is bounded to the maximum capacity of each machine's time window–as follows:

$$\hat{\mu}_d = e^{(\hat{\beta}_0 + \hat{\beta}_1 d + \hat{\beta}_2 d^2)} \tag{4}$$

$$Occ_d = \begin{cases} \mathcal{NB}\left(\hat{r}_d, \dfrac{\hat{r}_d}{\hat{r}_d + \hat{\mu}_d}\right) & \text{if overdispersion} \\ \mathcal{P}(\hat{\mu}_d) & \text{otherwise} \end{cases} \tag{5}$$

To model beam-matched machines, the user specifies the number of partial and complete beam-matched sets, constrained to a maximum of machines_qty/2. The generator then randomly creates that specific number of sets, with each set's size sampled randomly, constrained by the number of available machines as $\text{setLen} = randInt(2, \text{machines\_qty}/\text{sets\_qty})$.

Patient arrival generation. For both reference datasets, the number of patients arriving each day is modelled as a Poisson distribution. The mean λ is provided by the user and should be chosen within the ranges observed in the reference data: $\lambda \in [3.6, 5.6]$ for **ASUIT** and $\lambda \in [9, 10.7]$ for **Netwerk**–calculated as the minimum and maximum per-instance daily mean across all instances. The preferred and allowed machines selected for each protocol are randomly generated according to the proportions taken from the reference datasets analysis. Finally, each patient is assigned a priority, sampled according to the priority proportions defined by the user. Given a priority, a protocol is then sampled from a discrete distribution over protocols of that specific priority, where each protocol's probability is proportional to its observed frequency in the **Netwerk**

dataset instances. Protocol attributes are also taken from the **Netwerk** dataset and are applied to both datasets since the **ASUIT** protocols are not standardly defined–i.e., the physicians decide the treatment ad hoc, not following a database of standard treatments.

4.5 Generator parameters

The instance generator takes a command-line argument that contains the file with the parameters to drive the generation. Those parameters can be divided into three different categories: **dataset**, **machines**, and **patients** configurations.

Dataset configuration. Three dataset parameters control the overall process. The `dataset_title` is a string identifier used to name the output directory where the generated instances are stored. The `instances_number` specifies how many independent instances are created in a single run. Lastly, the `seed` parameter sets the random seed for reproducibility.

Machines configuration. To generate the machines, their occupation, and their compatibility (beam-matching), the generator takes several parameters as input. Firstly, the number of machines and of time windows is stated by `machines_number` and `time_windows_number` parameters. Then, the maximum capacity of each machine for each time window is stated by the `capacities` parameter. The `initial_occupation_percentage` (ω) parameter is used along with the maximum capacity to state the initial occupation $\hat{\beta}_0$ for each machine and time windows as stated in Equation (6). Moreover,`occupation_decay_velocity` (ν) controls how quickly the pre-existing occupation decreases over time, interpolating between a fast-decay coefficient $\beta_1^{\min}$ and a slow-decay coefficient $\beta_1^{\max}$ combined as described in Equation (7). Lastly, the minimum quantity of complete and partial beam-matched machine sets is stated by `min_beam_matched_sets` and `min_partial_beam_matched_sets`.

$$\hat{\beta}_0 = \ln(\omega \cdot \texttt{capacities}) \qquad (6) \qquad\qquad \hat{\beta}_1 = \nu \cdot \beta_1^{\min} + (1-\nu) \cdot \beta_1^{\max} \qquad (7)$$

Patients configuration. As far as the patient arrival generation is concerned, the generator uses two parameters. The `arrival_mean`, which is the expected average of patient arrival per day, is used in a Poisson distribution as the λ parameter from which the actual number of patients arriving each day is sampled. Finally, the `priorities_percentage` parameter is a probability vector that specifies the proportion of patients who belong to each priority class.

4.6 Solvers and Validator

The tool also includes a suite of solvers, such as a Mixed Integer Linear Programming (MILP) solver, a Simulated Annealing (SA) algorithm, and two heuristics. The details of the implementation of these solvers are described in [14]. The validator script takes the path to the *input* and *solution* files and checks the feasibility of the solution, as described by Section 3. This gives the user the ability to check the feasibility of the solutions found by the various solvers.

5 Evaluation

Generated Instances. We generated 7 new datasets (from now on referred to as `d_#` where `#` is the number of the dataset) with different parameters [5]. All the datasets are generated using the same seed (198743), and start from the same occupation percentage ($\omega = 70\%$); `d_1` contains 100 instances, while `d_2` to `d_7` contain 30 instances each. Datasets from `d_1` to `d_5` contain half of the instances generated with 7 patients arriving on average every day, and half with 10 patients, while `d_6` and `d_7` have, respectively, an average number of patients arriving of 10 and 5, for all instances. Datasets `d_1`, `d_3`, `d_4`, `d_5` have 6 machines, while `d_2` has 4 machines; for all these datasets, the machines are divided into 2 time windows with 240 minutes of capacity each. Datasets `d_6` and `d_7` try to mimic the **Netwerk** and the **ASUIT** reference datasets, respectively. In fact, `d_6` has 10 machines, and half of the instances are generated with 2 time windows of 240-minute capacity each, and the other half with 4 time windows of 120-minute capacity each. On the other hand, `d_7` has 4 machines, half instances with 2 time windows with capacity 240 and 210 minutes, and the other half with 4 time windows with capacity 120, 120, 135, and 75 minutes. Datasets from `d_1` to `d_3` have an occupation decay velocity ν set to 0.75, and beam-matched (BM) sets and partial beam-matched (PBM) sets both to 0; the other datasets have an occupation decay velocity set to 0.5, and BM sets and PBM sets both to 1, with the exception of `d_6` configured to generate 2 BM sets and 3 PBM sets. Finally, the patient priority percentages (from the most important to the least important) are [0.3, 0.3, 0.4] for `d_1` and `d_4`, [0.6, 0.2, 0.2] for `d_2`, `d_3`, and `d_5`, [0.75, 0.2, 0.05] for `d_6`, and [0.2, 0.1, 0.7] for `d_7`.

Instance Space Analysis. Since the reference datasets come from two really different facilities in terms of number of machines and daily patient arrivals, we decided to use Instance Space Analysis (ISA) through the software *Matilda* [17] to assess how they spread in the feature space, finding they form two different and really distant clusters as expected. The newly generated datasets were purposely created so as to fill the space between the two clusters. We solved the new instances with two of the solvers among the ones provided by our tool, i.e., MILP and SA, and we extracted 95 direct features (directly from the instances, based on simple statistical aggregations). The ISA selects 10 features and builds a projection matrix to map them to a 2-dimensional space, as follows:

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} 0.4924 & 0.3562 \\ -0.3678 & -0.5242 \\ 0.776 & 0.2218 \\ 0.3738 & 0.2447 \\ -0.7577 & -0.1519 \\ -0.3962 & 1.0747 \\ 0.187 & -0.6391 \\ -0.3649 & -0.0674 \\ -0.5144 & 0.7411 \\ -0.4718 & -1.0413 \end{pmatrix}^{\top} \cdot \begin{pmatrix} \text{patientsP1PerDayGini} \\ \text{patientsP2PerDayRange} \\ \text{patientsP3PerDayMedian} \\ \text{patientsP3PerDayStd} \\ \text{patientsP3PerDayGini} \\ \text{beamMatchedMachinesQ3} \\ \text{machineResidualCapacityPercentageByDayMin} \\ \text{machineResidualCapacityPercentageByDayRange} \\ \text{machineResidualCapacityPercentageByDayStd} \\ \text{allowedMachinesForPatientStd} \end{pmatrix} \tag{8}$$

[5] All the generated instances, along with the code, can be found at https://github.com/DIOL-UniTN/RTSP-Benchmark-Instance-Generator-Tool.git

Figure 1 presents the distribution of the different instances across the previously described 2-dimensional feature space. It can be seen that the two reference datasets constitute two clusters in opposite corners of the plot. It is worth noticing that the **ASUIT** dataset is split into "real" and "synthetic"; this is due to the fact that real data cannot be made public. For this reason, a synthetic, real-like dataset has been created to be made publicly available. Thanks to the ISA analysis, we can also verify that the synthetic version of the **ASUIT** dataset overlaps with the real one, making it safe to assume the two datasets are statistically similar (i.e., they are sampled from the same distribution). Datasets `d_1`, `d_2`, `d_3` cover another corner of the feature space, specifically the lower right one. Datasets `d_4` and `d_5` cover the centre of the space, while `d_6` and `d_7` mimic the distributions of the **Netwerk** and **ASUIT** reference datasets, respectively.

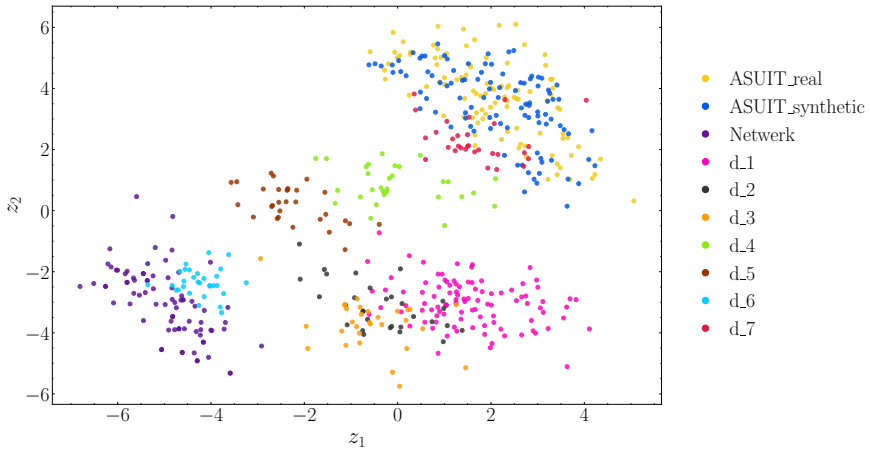


Fig. 1. A scatter plot visualizing how the different instances are spread in the 2-dimensional space found by multiplying the projection matrix by the processed feature vector.

6 Conclusion and Future Work

The RTSP is a relevant scheduling problem, which aims to ensure that timely treatments are performed and optimise patients' preferences and machine usage. Despite the problem being addressed in the existing literature, we found a lack of standardised and scalable available datasets and solvers. In this paper, we presented a Benchmark Instance Management Tool that can generate, solve, and validate new RTSP instances. The generator is constructed upon two reference datasets from real-world facilities of considerably different dimensions: a big cancer centre from Belgium [8] and a small radiotherapy unit from Italy (unpublished data). Those datasets have been statistically analysed to find the correct distributions and parameter ranges for the generation of new, realistic instances. The reference datasets' instances have been evaluated, along with some newly

generated ones, with the Instance Space Analysis (ISA) tool Matilda [17]. We extracted 95 direct features from the instances, solved them with MILP and SA solvers, and fed the results to the ISA tool, which mapped the 10 most relevant features into a 2-dimensional space. We analysed how the instances spread across the 2-dimensional space and concluded that the proposed generator is able to create real-world-like instances that can cover all the space between the instances obtained from two different-sized facilities and even explore other portions of the instance space. This makes the proposed Benchmark Instance Management Tool a significant asset in the RTSP domain, as it gives the opportunity to generate new instances, solve them with the given solvers or with new developed ones, and easily compare the results across all datasets. It could also be used to perform what-if analysis, e.g., aimed at understanding the effect of an increased number of patients, a varying number of machines, etc. Future works will focus on making the generator even further customisable, giving the users the possibility to add protocols, decide the specific sets of beam-matched machines, and preferred/allowed machines for protocols or patients.

Acknowledgments. This paper is based upon work from a scholarship supported by SPECIES (http://species-society.org).

References

1. Boonmee, C., Pisutha-Arnond, N., Chattinnawat, W., Muangwong, P., Nobnop, W., Chitapanarux, I.: Decision support system for radiotherapy patient scheduling: Thai cancer center case study. In: International Conference on Medical and Health Informatics. pp. 168–175 (2021)
2. Braune, R., Gutjahr, W.J., Vogl, P.: Stochastic radiotherapy appointment scheduling. Central European Journal of Operations Research pp. 1–39 (2022)
3. Burke, E.K., Leite-Rocha, P., Petrovic, S.: An integer linear programming model for the radiotherapy treatment scheduling problem. arXiv preprint arXiv:1103.3391 (2011)
4. Cares, J.P., Riff, M.C., Araya, I.: LS2R: A local search algorithm to solve scheduling radiotherapy problems. In: International Conference on Hybrid Intelligent Systems (HIS 2013). pp. 256–261. IEEE (2013)
5. Conforti, D., Guerriero, F., Guido, R.: Optimization models for radiotherapy patient scheduling. 4OR **6**, 263–278 (2008)
6. Emsamrit, N., Boonmee, C.: Mixed-Integer Linear programming for scheduling of radiotherapy patients. Engineering and Applied Science Research **51**(1), 106–116 (2024)
7. Frimodig, S., Enqvist, P., Carlsson, M., Mercier, C.: Comparing optimization methods for radiation therapy patient scheduling using different objectives. In: Operations Research Forum. vol. 4, p. 83. Springer (2023)
8. Frimodig, S., Enqvist, P., Kronqvist, J.: A Column Generation Approach for Radiation Therapy Patient Scheduling with Planned Machine Unavailability and Uncertain Future Arrivals (2023), arXiv:2303.10985
9. Higgins, G., Shah, K., Horne, P., Gilham, M., Bloomfield, D., et al.: The timely delivery of radical radiotherapy: guidelines for the management of unscheduled treatment interruptions. Londres: The Royal College of Radiologists (2019)

10. Jacquemin, Y., Marcon, E., Pommier, P.: Towards an improved resolution of radiotherapy, scheduling. In: 2010 IEEE workshop on health care management (WHCM). pp. 1–6. IEEE (2010)
11. Legrain, A., Fortin, M.A., Lahrichi, N., Rousseau, L.M.: Online stochastic optimization of radiotherapy patient scheduling. Health Care Management Science **18**(2), 110–123 (2015)
12. Maschler, J., Riedler, M., Stock, M., Raidl, G.R.: Particle therapy patient scheduling: First heuristic approaches. In: International Conference on the Practice and Theory of Automated Timetabling. pp. 223–244 (2016)
13. Pham, T.S., Rousseau, L.M., De Causmaecker, P.: A two-phase approach for the Radiotherapy Scheduling Problem. Health Care Management Science pp. 1–17 (2022)
14. Rambaldi Migliore, C.C., Stanicel, D., Musliu, N., Iacca, G., Roveri, M.: Comparing Optimization Models for Radiotherapy Scheduling. arXiv preprint (2026)
15. Roomi, M.W.: Radiation Therapy for Cancer Treatment. Journal of Otolaryngology Research & Reports **2**, 1–10 (2023)
16. Saure, A., Patrick, J., Tyldesley, S., Puterman, M.L.: Dynamic multi-appointment patient scheduling for radiation therapy. European Journal of Operational Research **223**(2), 573–584 (2012)
17. Smith-Miles, K., Muñoz, M.A.: Instance space analysis for algorithm testing: Methodology and software tools. ACM Computing Surveys **55**(12), 1–31 (2023)
18. Vieira, B., Demirtas, D., van de Kamer, J.B., Hans, E.W., Jongste, W., van Harten, W.: Radiotherapy treatment scheduling: Implementing operations research into clinical practice. PLOS ONE **16**(2), e0247428 (2021)
19. Vieira, B., Demirtas, D., van de Kamer, J.B., Hans, E.W., Rousseau, L.M., Lahrichi, N., van Harten, W.H.: Radiotherapy treatment scheduling considering time window preferences. Health Care Management Science **23**, 520–534 (2020)
20. Vieira, B., Hans, E.W., van Vliet-Vroegindeweij, C., van de Kamer, J., van Harten, W.H.: Operations research for resource planning and -use in radiotherapy: a literature review. BMC Medical Informatics Decis. Mak. **16**, 149:1–149:11 (2016)
21. Vogl, P., Braune, R., Doerner, K.F.: A multi-encoded genetic algorithm approach to scheduling recurring radiotherapy treatment activities with alternative resources, optional activities, and time window constraints. In: International Conference on Computer Aided Systems Theory. pp. 373–382. Springer (2017)
22. Vogl, P., Braune, R., Doerner, K.F.: Scheduling recurring radiotherapy appointments in an ion beam facility: Considering optional activities and time window constraints. Journal of Scheduling **22**(2), 137–154 (2019)
23. Wild, C.P., Weiderpass, E., Stewart, B.W.: World cancer report. Radiotherapy and oncology : journal of the European Society for Therapeutic Radiology and Oncology (2020)

Modular Constraint Injection from Natural Language for CP-Based Scheduling

Florian Strohm[1], Patrick Wagner[1], Jannik Schwab[1], and Marco F. Huber[1]

Fraunhofer Institute for Manufacturing Engineering and Automation (IPA), Stuttgart, Germany
{florian.strohm, patrick.wagner, jannik.schwab, marco.huber}@ipa.fraunhofer.de

Abstract. Clinic scheduling systems built on constraint programming produce high-quality schedules, but adapting them to frequently changing operational rules remains a bottleneck: every new constraint, from staff availability changes to patient-specific restrictions, requires a developer to translate domain knowledge into solver code. We present an LLM-based agent that lets non-technical clinic staff add scheduling constraints by describing them in natural language. New constraints are injected as modular *plugins* with a standardized interface, preserving the integrity of the core scheduling model. The agent autonomously resolves entity references against the clinic database and verifies generated code through self-designed test scenarios before deployment. On a benchmark of 234 scheduling constraints across nine operationally relevant categories, the agent achieves 90.2% accuracy versus 70–82% for alternative approaches. Ablation reveals that database search and iterative self-testing are complementary, with their combination (+20.1 pp) exceeding the sum of individual contributions (+16.2 pp). The improvements hold across three LLMs, with a consistent +13–19 pp uplift over baselines.

Keywords: staff scheduling · constraint programming · large language models · agentic AI · code generation

1 Introduction

Staff scheduling in healthcare is shaped by a dense web of constraints that evolve continuously: infection-control protocols change, staff availability shifts week to week, and patients bring unique requirements [11,7]. Constraint programming (CP) has long been a natural framework for such problems [26], and modern solvers such as Google OR-Tools CP-SAT [24] routinely optimize schedules under hundreds of interacting constraints. Yet a persistent gap separates the people who *understand* scheduling rules from those who can *encode* them. Freuder [12] famously termed bridging this gap the "Holy Grail" of CP, later proposing conversational modelling as a path toward it [13]. In practice, every rule change ("Dr. Martinez cannot work Fridays," "patients over 75 should only be seen in

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

ground-floor rooms") must be communicated to a developer, formalized, tested, and deployed, introducing delay and miscommunication. Our work with rehabilitation clinics confirmed that even well-maintained CP systems require frequent, developer-dependent constraint updates. Large language models (LLMs) suggest a path forward: they can generate code [8,20], reason over multi-step problems [38], and use tools autonomously [39]. However, translating a natural language scheduling constraint into correct CP code requires *entity resolution* (mapping "the wheelchair patient in Building C" to a database record) and *semantic precision* (avoiding constraints that are syntactically valid but logically incorrect).

We frame our task as *incremental constraint injection*: given an already-deployed CP scheduling model and a new constraint in natural language, produce executable code that enforces the constraint *without modifying* existing logic. This differs from complete model generation [35,32]: the base model already handles core objectives, and new constraints must compose safely. Our system addresses this through (1) a *plugin architecture* injecting constraints as isolated functions with a standardized interface, and (2) a *tool-augmented LLM agent* combining database search with self-designed test scenarios.

Our contributions are as follows:

1. A **plugin-based constraint injection architecture** for CP-SAT models with modular, safe, and reversible constraint management (Sec. 3.2).
2. A **tool-augmented LLM agent** with database search and self-testing for autonomous constraint generation and verification (Sec. 3.3).
3. A **benchmark of 234 scheduling constraints** spanning nine categories with oracle-based evaluation (Sec. 4).[1]
4. An **empirical evaluation** showing 90.2% accuracy vs. 70–82% for alternatives, with complementary benefits from search and self-testing (Sec. 5).

2 Related Work

Staff scheduling and CP. Staff scheduling and rostering have been studied extensively [11,7]. CP has been a mainstay approach, with recent applications including medical appointment sequencing [2] and machine learning–CP hybrids for healthcare scheduling [4]. These efforts build or improve scheduling *algorithms*; we target the complementary problem of enabling domain experts to *customize* an existing system via natural language.

Natural language and CP. Tsouros et al. [35] frame the "Holy Grail 2.0" vision of translating natural language into constraint models, and several systems pursue it: retrieval-augmented in-context learning [21], benchmarks showing that self-verification helps [22], ReAct-style agents reaching high accuracy [32], solver

[1] Code, benchmark, and artifacts will be released upon publication.

bridges via the Model Context Protocol [31], and fine-tuning with constraint-aware retrieval [27]. Song and Cohen [29] note that *paraphrasing* the input degrades LLM accuracy, a finding directly relevant to our setting, where one constraint type recurs for many entities. Other lines embed an LLM inside the solver so propagation guides generation [5]. Adjacent efforts target sub-steps or related tasks: named-entity extraction of optimization components [10], interactive meeting scheduling [18], SAT-backed explainable scheduling [36], template-matched care-worker constraints [30], expert-guided MILP generation [19], and streamliner synthesis [37]. Most of these start from a full problem description to produce a complete formulation, target solver performance, or address only an extraction sub-step; none address *incremental constraint injection* into an already-deployed model, a task requiring entity resolution against a live database. Our work fills this gap with a plugin architecture and a self-testing loop that also improves within-pattern consistency.

Rule-based and high-level modelling languages. A long line of work lowers the barrier to changing an optimisation model without editing solver code. Domain-specific rule languages, exemplified by airline crew-scheduling systems such as Carmen [16], let planners express rostering rules in a controlled vocabulary over a fixed optimisation core; high-level constraint languages (MiniZinc [23], Essence [14], CPMpy [15], and the XCSP3 format [6]) provide declarative, solver-independent comprehensions, and the global constraint catalogue [3] standardises reusable high-level relations such as `alldifferent` and `cumulative`. These approaches are more expressive and more checkable at the modelling layer than ours: our generated constraints sit at CP-SAT's Boolean-linear level rather than using such global constructs (Section 3.4). Our contribution is orthogonal and complementary. Rather than defining a notation the user must learn and bind to correctly-spelled data, the agent resolves *informal* natural-language references against a live database at authoring time, e.g. mapping "the wheelchair patient in Building C" to rows, or matching a request for a "Spanish-speaking" staff member against the recorded `staff_languages`. A high-level modelling language could in fact serve as the agent's *generation target*, pairing schema-grounded entity resolution with compiler-checked, global-constraint output; we see this as a promising direction.

LLM-based optimization and self-debugging. The NL4Opt shared task [25] established an early benchmark of 1,101 LP problems for NL-to-optimization translation. In mathematical programming, OptiMUS [1] and others [33,40] generate optimization models from natural language. ConstraintBench [34] finds only 65% feasibility for direct LLM solutions, motivating solver-in-the-loop approaches. Iterative self-debugging [28,9] has proven effective for code generation; our agent applies similar principles with domain-specific feedback from CP-SAT feasibility.

3 System Design

3.1 Scheduling Domain

Our system assigns patient treatment needs to treatment slots in a clinic scheduling setting. A configurable *clinic generator* creates realistic synthetic instances (parameterized by staff, rooms, patients, treatment types, horizon, and target utilization) with values informed by real-world rehabilitation clinic operations. Our benchmark uses a medium-sized instance: 100 patients with conditions and allergies, 72 treatment types, 1,425 needs, 1,000 slots across 7 days, 50 staff with certifications, and 30 rooms across 5 buildings. The base CP-SAT model is deliberately minimal. Let N be the set of treatment needs and S the set of slots, and let $C \subseteq N \times S$ be the *compatibility* relation $C = \{(n, s) : \text{treatment}(s) = \text{treatment}(n)\}$ that pairs each need with the slots offering its treatment. We introduce one Boolean decision variable per compatible pair,

$$x_{n,s} \in \{0,1\}, \quad (n,s) \in C, \qquad x_{n,s} = 1 \iff \text{need } n \text{ is assigned to slot } s, \quad (1)$$

and solve

$$\begin{aligned} \max \quad & \textstyle\sum_{(n,s)\in C} x_{n,s} && && (2)\\ \text{s.t.} \quad & \textstyle\sum_{s:\,(n,s)\in C} x_{n,s} \le 1 && \forall\, n \in N, && (3)\\ & \textstyle\sum_{n:\,(n,s)\in C} x_{n,s} \le \text{cap}(s) && \forall\, s \in S. && (4) \end{aligned}$$

Constraint (3) schedules each need at most once (leaving a need unscheduled is feasible), and (4) respects each slot's capacity $\text{cap}(s)$. Under (3) the objective (2) equals the number of scheduled needs, so the solver maximizes coverage. This two-family model together with structural compatibility is the *entire* deployed base model: operational rules such as staff certification, room equipment, patient allergies, timing windows, and workload fairness are absent from it and enter only as injected plugins (Section 3.2). We fix the solver's random seed and run it single-threaded, making every solve deterministic.

3.2 Plugin Architecture

New constraints are injected as *plugins*: self-contained Python functions that extend the base model without modifying its objective or existing constraints. A plugin posts additional relations over the existing assignment variables; it never alters the objective, removes a base constraint, or rebinds an existing variable. All plugins share the fixed signature in Listing 1.1.

A plugin may introduce its own auxiliary variables (for example to count occurrences or to reify a condition), but because it only *adds* relations over the shared variables $x_{n,s}$, the set of admissible assignments can only shrink when projected onto those variables: $\pi(F') \subseteq \pi(F)$, where π projects the feasible region onto $\{x_{n,s}\}$. This yields two properties: *(i) monotone composition*, where each accepted plugin further constrains the schedule without relaxing earlier ones,

Listing 1.1. Standardized plugin interface. A plugin posts `model.Add(...)` relations over the shared `assign` variables; `context` exposes the database read-only.

```
def add_constraint(model, assign, needs_by_id, slots_by_id,
    context):
    # model:       the live CpModel being extended
    # assign:      {(need_id, slot_id): BoolVar}  -- the x_{
        n,s} above
    # needs_by_id, slots_by_id: need / slot attribute
        lookups
    # context:     read-only tables (patients, staff, rooms,
    #              certifications, allergies, equipment,
        travel times, ...)
    ...  # may add plugin-local auxiliary variables, then
        post constraints
```

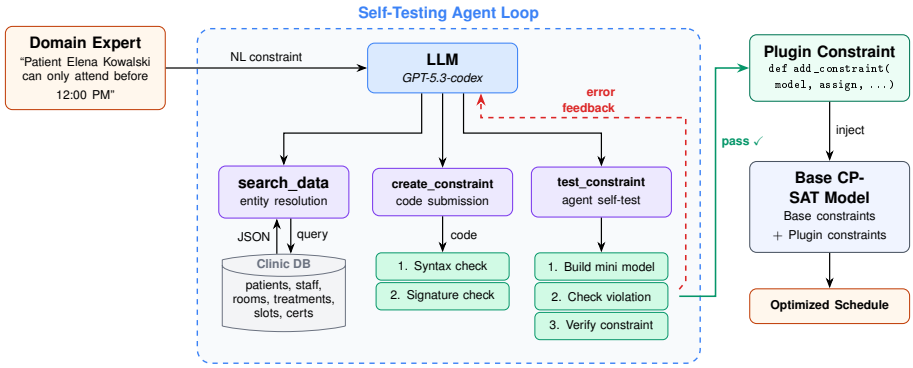


Fig. 1. System overview. A domain expert describes a constraint in natural language. The agent resolves entities via database search, generates code, verifies it through self-designed tests, and injects validated constraints as read-only plugins.

so rules compose through the shared feasible region even though their code is written independently; and *(ii) code-level isolation*, where plugins share only read-only data and the variable handles, so one plugin's code cannot corrupt the core model or another plugin. The design thus provides *modularity* (isolated constraints), *safety* (the core model cannot be overwritten), *reversibility* (constraints toggle on and off), and *auditability* (each function is independently reviewable).

3.3 Agent Architecture

Our agent (Figure 1) is a LangGraph [17] state machine iterating between an LLM and three tools for up to 10 rounds (over 95% converge within 5; the limit prevents costly unproductive loops):

`search_data(query_type, term)` searches the clinic database by entity type and returns up to 20 matching records as JSON, enabling entity resolution (e.g., "Dr. Martinez" → `staff_id 15`).

`create_constraint(code)` validates syntax and verifies that the code defines a callable `add_constraint` function.

`test_constraint(test_code)`, the key differentiator, lets the agent submit a self-designed test: a minimal CP-SAT model that checks (1) that a violation exists without the constraint and (2) that the violation is fixed with it. Tests run sandboxed with a 30-second timeout.

The system prompt provides the function signature, data schema, four CP-SAT code patterns, and seven correctness rules. In a typical loop, the agent generates code, discovers a string/integer type mismatch through `test_constraint`, fixes the comparison, and re-tests successfully.

3.4 From Natural Language to CP Rules

To make the translation concrete, consider the request *"junior-level staff members cannot be assigned to any surgical treatment."* This is not a change to a stored field: the base model contains no notion of seniority or treatment category, and no single row expresses "for every junior staff member and every surgical slot." The agent resolves `junior` against the `staff` table and `surgical` against the `treatments` table, then posts, for every compatible pair routed through a junior-staffed surgical slot, the relation

$$x_{n,s} = 0 \quad \forall\,(n,s) \in C : \mathrm{junior}\big(\mathrm{staff}(s)\big) \wedge \mathrm{surgical}\big(\mathrm{treatment}(s)\big). \tag{5}$$

The generated plugin (Listing 1.2, Appendix A) builds the `junior` and `surgical` id sets by a two-table scan, then posts Eq. (5) for every matching compatible pair. The Python loop is model *construction*, not search: it declaratively adds one linear relation per matching variable, much as a `forall` comprehension would in a higher-level modelling language. A second, aggregation-style example appears in the same appendix gallery.

4 Benchmark

4.1 Benchmark Construction

We construct a benchmark of 234 scheduling constraints designed to test a system's ability to translate diverse natural language requests into correct CP-SAT code. The constraints cover nine operationally relevant categories that reflect the rules clinic managers, head nurses, and schedulers encounter in practice: Temporal (48), Staff (38), Complex (34), Resource (31), Spatial (24), Conditional (18), Preference (17), Fairness (17), and Relational (7). For constraint types where surface-level variation is natural and frequent, such as patient-specific day restrictions ("Patient X cannot come on Mondays" vs. "Patient Y cannot

come on Thursdays") or staff daily caps, we include multiple instances sharing the same logical template but differing in entity names, parameter values, or phrasing. These *within-pattern variants* serve a dual purpose: they reflect the repetitive nature of real scheduling requests (a clinic manager will issue the same type of constraint for many different patients), and they measure *consistency*, i.e., whether a system that solves one instance of a pattern can reliably solve another. Of the 234 tests, 129 belong to 54 multi-instance pattern clusters (2–4 tests each), while 105 are unique patterns. The benchmark consists of two tiers. The first 200 constraints were initially generated using an ensemble of LLMs from a detailed prompt specifying the scheduling domain and schema. From approximately 400 candidates, we selected and refined 100 core constraints that are concrete, testable, and span all categories, then created 100 variations by rephrasing, changing parameter values, or targeting different entities. The remaining 34 constraints (tests 201–234) form a *complex tier* that stress-tests multi-step reasoning: multi-hop entity resolution, implicit/descriptive references (e.g. "the youngest surgeon"), cross-table joins over 3+ tables, compound logic with exception clauses, and tricky boundary- or domain-knowledge formulations.

4.2 Test Protocol

Each test follows a standardized three-step protocol: (1) **Baseline check**: run the scheduler *without* the constraint and verify that at least one violation exists in the resulting schedule; if no violation exists, the test is skipped as inapplicable to the current data instance. (2) **Constraint injection**: add the generated constraint to the scheduling model. (3) **Verification**: re-run the scheduler and assert that *all* assignments now satisfy the constraint oracle. This protocol ensures every test is meaningful: the constraint must actually change the schedule to pass. All 234 tests use *dynamic entity resolution*: rather than hardcoding specific patient or staff names, each test searches for entities matching required characteristics at runtime, making the benchmark *instance-independent*: the same tests work across different clinic data instances produced by the generator. Each test implements its verification oracle as a standalone Python assertion function that checks every assignment in the returned schedule, cross-referencing the scheduling database. A two-stage data pipeline (broad-coverage generation plus idempotent patching) ensures all 234 tests have baseline violations on three verified instances, enabling cross-instance robustness evaluation.

Beyond violation elimination. The protocol above is a *soundness* check: it confirms a constraint removes its target violation, but not that it leaves unrelated assignments intact; a constraint can pass while *over-restricting* the feasible region. We therefore add a *non-interference* check for exclusion-type rules: every baseline assignment that does *not* match the rule's antecedent must survive injection. The rule "patients with a life-threatening contrast-dye allergy must avoid CT-scan rooms" illustrates the gap. It affects two patients; the correct encoding excludes them from the three CT-scan rooms, but the agent's generated plugin mis-keys the equipment table (reading `equipment_name` instead of

`equipment_type`) and falls back to excluding all six imaging rooms. Both encodings satisfy the violation-elimination oracle (no affected patient occupies a CT-scan room), yet the over-broad plugin also removes both patients from the three *safe* non-CT imaging rooms, forbidding 38 additional (need, slot) pairs and scheduling one fewer need overall (objective 1421 vs. 1422). The non-interference check flags this; the soundness oracle does not. For aggregate rules, where the minimal coverage loss is itself an optimization, we instead report the objective delta as a diagnostic rather than a guarantee.

5 Experiments

5.1 Setup

We evaluate six configurations: (1) **Agent** (full system), (2) **Agent (test)** (self-testing only, no database), (3) **Agent (search)** (database only, no self-testing), and three non-agentic baselines defined below: (4) **Sampling**, (5) **Few-shot**, and (6) **Zero-shot**. All use GPT-5.3-codex (temperature 0, except Sampling), run three times for variance estimation. Each constraint is solved with CP-SAT (OR-Tools 9.15) using a single search worker and a fixed random seed (1609), making each solve deterministic; each test additionally runs under a 300-second wall-clock limit. Three data instances (seeds 1609, 2024, 7777) enable cross-instance evaluation.

Prompting baselines. The three non-agentic baselines share the agent's system prompt (function signature, data schema, four CP-SAT patterns, seven correctness rules) and emit a complete plugin in a single forward pass, with no database tool and no solver feedback. *Zero-shot* issues one generation at temperature 0 with no examples. *Few-shot* prepends five fixed, hand-written natural-language-to-code exemplars (among them a domain restriction, a linear cardinality bound, and a reified building count) and then generates once at temperature 0. *Sampling* draws N=5 independent generations at temperature 0.7 and keeps the first that passes static validation (it compiles and defines a callable `add_constraint`), a diversity strategy rather than best-of-N by quality, which is why it falls between few-shot and the agent.

5.2 Results

The agent achieves 90.2% accuracy (211/234), a 20.1 pp improvement over zero-shot (Table 1). All improvements are highly significant by McNemar's test ($p < 0.001$; Cohen's $h = 0.49$ vs. few-shot). Of 234 tests, 133 are "easy" (all pass) and 13 "hard" (all fail), leaving 88 discriminating tests where the agent achieves 88.6% vs. 62.5–67.0% for the next-best systems.

Table 1. Overall results (n=234) and discriminating tests (88 tests where systems differ).

System	Accuracy	Discrim. (n=88)
Agent	**90.2%**	**88.6%**
Agent (test)	82.1%	67.0%
Sampling (N=5)	80.3%	62.5%
Agent (search)	74.4%	46.6%
Few-shot	71.4%	38.6%
Zero-shot	70.1%	35.2%

Table 2. Accuracy (%) by category (selected); category size n in parentheses.

	Temporal (n=48)	*Complex* (n=34)	*Staff* (n=38)	*Resource* (n=31)	*Conditional* (n=18)	*Fairness* (n=17)
Agent	**93.8**	**91.2**	**81.6**	**93.5**	**100.0**	**76.5**
Agent (test)	85.4	58.8	**81.6**	87.1	72.2	82.4
Sampling	83.3	55.9	76.3	90.3	83.3	82.4
Zero-shot	77.1	52.9	73.7	74.2	72.2	41.2

Ablation: search and self-testing are complementary. Measured against zero-shot, self-testing alone contributes +12.0 pp (82.1%) and database search alone +4.3 pp (74.4%). Yet each mechanism is far more effective in the presence of the other: self-testing adds +15.8 pp on top of search (74.4→90.2%), and search adds +8.1 pp on top of self-testing (82.1→90.2%). The combined +20.1 pp therefore exceeds the sum of the two standalone contributions (+16.2 pp), indicating that the mechanisms reinforce each other.

Category analysis. Our per-category breakdown (Table 2; selected categories) shows where capabilities matter most. *Complex constraints* (91.2% vs. 53–59%) show the starkest separation: these 34 tests require multi-hop entity resolution and compound logic, and systems without database search perform far worse. *Conditional constraints* reach 100% (vs. 72–89%) by combining search with self-testing. *Temporal* and *resource* constraints (93.5–93.8%) benefit from self-testing catching type errors and infeasibility. The search-only agent underperforms few-shot on several categories (e.g., Temporal: 70.8% vs. 85.4%), suggesting database results can introduce distracting context when the agent lacks execution feedback; self-testing is needed to realize the benefit of entity resolution.

Error analysis. Of 23 non-passing tests, 7 are solver timeouts (all fairness constraints with $O(|\text{staff}| \times |\text{days}|)$ aggregation) and 16 logic errors. Root causes:

missing cross-table joins (5/16), incorrect value comparisons (5/16), wrong entity resolution (3/16), and constraints that are semantically too weak, passing self-tests but not fully capturing user intent (3/16). The last category is clinically concerning: a constraint intended to *exclusively* assign morning slots might merely add a soft *preference*, a semantic gap that our self-testing cannot detect.

Robustness and generalization. The agent is stable across runs (90.2±0.9%; 218/234 tests identical), data instances (89.3±0.9% across three seeds), and model families: GPT-5.1-Mini and Kimi-K2.5 (open-weights) show a consistent +13–19 pp uplift over baselines. It also generalizes within constraint *patterns*: across the 54 multi-instance clusters it solves all instances of a pattern consistently in 87.0% of cases (47/54) versus 72.2% (39/54) for zero-shot. Overhead is moderate: it passes zero-shot within 60 s per constraint and sampling at 120 s, with diminishing returns beyond 180 s.

6 Discussion

Why the agent loop works. The agent's 20.1 pp improvement over zero-shot generation stems from two complementary error-correction mechanisms, each targeting a different failure mode.

Self-testing catches code-level errors. A failed self-test pinpoints the fault (a type mismatch, off-by-one error, or missing edge case), which the agent fixes iteratively, usually within 2–3 rounds. The benefit grows with complexity: complex constraints (tests 201–234) combine independently-failing sub-conditions, explaining the agent's 91.2% there versus 53–59% for single-pass systems that must get every sub-condition right at once.

Database search resolves data-level errors. Constraints reference entities, field values, and cross-table relationships absent from the constraint text, so without data access the LLM must guess, e.g. testing `insurance_type == "Public"` when the database stores `"public"`, silently constraining no one. The agent instead queries the database for actual values before generating code.

Practical implications. The plugin architecture matters for deployed scheduling tools: where constraints span staff qualifications, equipment, and patient conditions, adding them without risking the core logic is operationally important, and different stakeholders can add rules independently with reversibility and audit trails. For clinical deployment we recommend three safeguards: a *dry-run mode* showing a before/after schedule diff so experts can validate effects without reading code; audit trails with rollback; and human approval before activation (decision support, not automation).

Limitations. The self-testing loop verifies that the augmented model remains *feasible* but does not guarantee *semantic correctness*: a constraint that is too weak (e.g., restricting only Monday instead of all weekdays) may pass the feasibility test while not capturing the user's intent. Three of 16 logic errors fall into this

"too weak constraint" category. Conversely, an overly strict constraint can cause harm by excluding valid assignments, e.g., denying a patient necessary treatment slots. Because each plugin can only shrink the feasible region projected onto the assignment variables (Section 3.2), violation-elimination alone cannot detect this, and the agent's own self-test shares the blind spot. The benchmark-side non-interference check of Section 4.2 catches such over-restriction for exclusion-type rules, but an in-loop minimality check for the deployed agent (and coverage auditing for aggregate constraints, where minimality is itself an optimization) remains future work. We evaluate on clinic scheduling only; generalization to other scheduling domains (nurse rostering, university timetabling, job-shop scheduling) requires further study, though the plugin interface pattern is domain-agnostic in principle. The scheduling data is synthetically generated with clean naming; real-world data may introduce messier entity names, abbreviations, and missing fields. Finally, our benchmark is single-turn; although our live demonstrator supports multi-turn clarification (e.g., disambiguating a shared surname), neither this nor an end-to-end user study with clinic staff is yet formally evaluated.

7 Conclusion

We presented an LLM-based agent for translating natural language scheduling constraints into executable CP-SAT code, making deployed scheduling systems accessible to non-technical domain experts. Our plugin architecture ensures that generated constraints compose safely with existing scheduling logic, providing modularity, safety, reversibility, and auditability, properties essential for operational scheduling environments. Evaluation on 234 scheduling constraints spanning nine operationally relevant categories yields two principal findings. First, the agent achieves 90.2% accuracy versus 70–82% for five alternative approaches, with the gap widening to 88.6% versus 35–67% on 88 discriminating tests ($p < 0.001$ by McNemar's test). The advantage is most pronounced on 34 complex tests requiring multi-hop reasoning: 91.2% versus 53–59% for baselines. Second, ablation shows self-testing and data access are complementary: together they yield +20.1 pp, exceeding the sum of their standalone gains. Beyond the agent itself, our benchmark of 234 constraints with oracle-based evaluation provides a reusable testbed for future work on LLM-driven constraint generation in scheduling. Cross-model evaluation on three LLMs, including an open-weights model, confirms that the benefit generalizes across model families (+13–19 pp uplift). Future work includes extending the plugin architecture to other scheduling domains (nurse rostering, university timetabling, job-shop scheduling); supporting constraint *modification* and *retraction*, since the current system only adds constraints; lifting generated constraints to higher-level global-constraint forms; and adding multi-turn user interaction and soft-constraint generation, directions the plugin interface can accommodate but that we have not yet implemented or benchmarked.

Use of AI Assistance. The constraint-generation agent studied here is itself LLM-based (GPT-5.3-codex, with GPT-5.1-Mini and Kimi-K2.5 in the cross-model experiments). Large language models also assisted the authors: the benchmark's natural-language constraints were proposed by an LLM ensemble and selected by the authors, and the verification oracles and experiment code were implemented collaboratively through agentic coding under author review; Claude (Anthropic) additionally supported research discussions and manuscript editing. No experimental result, table entry, or statistical test was produced by an LLM; all reported numbers come from the deterministic OR-Tools harness run by the authors, who take full responsibility for the paper.

References

1. AhmadiTeshnizi, A., Gao, W., Udell, M.: OptiMUS: Scalable optimization modeling with (MI)LP solvers and large language models. In: Proc. ICML (2024)
2. Assaf, G., Löffler, S., Hofstedt, P.: Constraint-based optimization for scheduling medical appointments. In: Proc. ICAART. vol. 3, pp. 94–103. SCITEPRESS (2025)
3. Beldiceanu, N., Carlsson, M., Demassey, S., Petit, T.: Global constraint catalogue: Past, present and future. Constraints **12**(1), 21–62 (2007)
4. Ben Said, A., Mouhoub, M.: Machine learning and constraint programming for efficient healthcare scheduling. arXiv preprint arXiv:2409.07547 (2024)
5. Bonlarron, A., Régin, F., De Maria, E., Régin, J.C.: Large language model meets constraint propagation. In: Proc. IJCAI. pp. 10036–10044 (2025)
6. Boussemart, F., Lecoutre, C., Piette, C.: XCSP3: An integrated format for benchmarking combinatorial constrained problems. arXiv preprint arXiv:1611.03398 (2016)
7. Burke, E.K., De Causmaecker, P., Vanden Berghe, G., Van Landeghem, H.: The state of the art of nurse rostering. J. Scheduling **7**(6), 441–499 (2004)
8. Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.d.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al.: Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374 (2021)
9. Chen, X., Lin, M., Schärli, N., Zhou, D.: Teaching large language models to self-debug. In: Proc. ICLR (2024)
10. Dakle, P.P., Kadioğlu, S., Uppuluri, K., Politi, R., Raghavan, P., Rallabandi, S., Srinivasamurthy, R.: Ner4Opt: Named entity recognition for optimization modelling from natural language. In: Proc. CPAIOR. pp. 299–319. Springer (2023)
11. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Sier, D.: Staff scheduling and rostering: A review of applications, methods and models. Eur. J. Oper. Res. **153**(1), 3–27 (2004)
12. Freuder, E.C.: In pursuit of the holy grail. Constraints **2**(1), 57–61 (1997)
13. Freuder, E.C.: Conversational modeling for constraint satisfaction. In: Proc. AAAI. vol. 38, pp. 22592–22597 (2024)
14. Frisch, A.M., Harvey, W., Jefferson, C., Martínez-Hernández, B., Miguel, I.: Essence: A constraint language for specifying combinatorial problems. Constraints **13**(3), 268–306 (2008)
15. Guns, T.: Increasing modeling language convenience with a universal n-dimensional array: CPpy as a python-embedded example. In: Proc. 18th Workshop on Constraint Modelling and Reformulation (ModRef) at CP (2019)

16. Kohl, N., Karisch, S.E.: Airline crew rostering: Problem types, modeling, and optimization. Annals of Operations Research **127**, 223–257 (2004)
17. LangChain: LangGraph: Building stateful, multi-actor applications with LLMs (2024)
18. Lawless, C., Schoeffer, J., Le, L., Rowan, K., Sen, S., St. Hill, C., Suh, J., Sarrafzadeh, B.: I want it that way: Enabling interactive decision support using large language models and constraint programming. ACM Trans. Interact. Intell. Syst. **14**(3), 1–32 (2024)
19. Li, Q., Zhang, L., Mak-Hau, V.: An LLM-powered MILP modelling engine for workforce scheduling guided by expert knowledge. arXiv preprint arXiv:2511.02364 (2025)
20. Li, R., Allal, L.B., Zi, Y., Muennighoff, N., Kocetkov, D., Mou, C., Marone, M., Akiki, C., Li, J., Chim, J., et al.: StarCoder: May the source be with you! In: arXiv preprint arXiv:2305.06161 (2023)
21. Michailidis, K., Tsouros, D., Guns, T.: Constraint modelling with LLMs using in-context learning. In: Proc. CP (2024)
22. Michailidis, K., Tsouros, D., Guns, T.: DCP-Bench-Open: Evaluating LLMs for constraint modelling of discrete combinatorial problems. arXiv preprint arXiv:2506.06052 (2025)
23. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: MiniZinc: Towards a standard CP modelling language. In: Proc. Principles and Practice of Constraint Programming (CP). LNCS, vol. 4741, pp. 529–543. Springer (2007)
24. Perron, L., Furnon, V.: OR-Tools. In: Google Operations Research Tools (2023)
25. Ramamonjison, R., Yu, T., Li, R., et al.: NL4Opt competition: Formulating optimization problems based on their natural language descriptions. In: Proc. NeurIPS Competitions Track, PMLR. vol. 220, pp. 189–203 (2023)
26. Rossi, F., Van Beek, P., Walsh, T.: Handbook of Constraint Programming, Foundations of Artificial Intelligence, vol. 2. Elsevier (2006)
27. Shi, W., Liu, M., Zhang, W., Shi, L., Jia, F., Ma, F., Zhang, J.: ConstraintLLM: A neuro-symbolic framework for industrial-level constraint programming. In: Proc. EMNLP. pp. 15999–16019 (2025)
28. Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., Yao, S.: Reflexion: Language agents with verbal reinforcement learning. In: Proc. NeurIPS. vol. 36 (2023)
29. Song, Y., Cohen, E.: Do LLMs understand constraint programming? Zero-Shot constraint programming model generation using LLMs. In: Proc. LION. Lecture Notes in Computer Science, vol. 15744. Springer (2025)
30. Suenaga, K., Furuta, T., Ono, S.: A study on constraint extraction and exception exclusion in care worker scheduling. Artif. Life Robot. (2025)
31. Szeider, S.: Bridging language models and symbolic solvers via the model context protocol. In: Proc. SAT. pp. 30:1–30:12 (2025), arXiv:2501.00539
32. Szeider, S.: CP-Agent: Agentic constraint programming. arXiv preprint arXiv:2508.07468 (2025), to appear at AGENT@ICSE 2026
33. Thind, R., Sun, Y., Liang, L., Yang, H.: OptimAI: Optimization from natural language using LLM-powered AI agents. arXiv preprint arXiv:2504.16918 (2025)
34. Tso, J., Schmittou, P., Huynh, Q., Hutchins, J.: ConstraintBench: Benchmarking LLM constraint reasoning on direct optimization. arXiv preprint arXiv:2602.22465 (2026)
35. Tsouros, D., Verhaeghe, H., Kadioğlu, S., Guns, T.: Holy grail 2.0: From natural language to constraint models. arXiv preprint arXiv:2308.01589 (2023)
36. Vasileiou, Y., Yeoh, W.: TRACE-CS: A synergistic approach to explainable course scheduling using LLMs and logic. In: Proc. AAAI (2025)

37. Voboril, F., Peruvemba Ramaswamy, V., Szeider, S.: StreamLLM: Enhancing constraint programming with large language model-generated streamliners. In: Proc. NSE@ICSE (2025), arXiv:2408.10268
38. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D.: Chain-of-thought prompting elicits reasoning in large language models. Proc. NeurIPS **35**, 24824–24837 (2022)
39. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., Cao, Y.: ReAct: Synergizing reasoning and acting in language models. In: Proc. ICLR (2023)
40. Zhang, B., Luo, P., Yang, G., Soong, B.H., Yuen, C.: OR-LLM-Agent: Automating modeling and solving of operations research optimization problems with reasoning LLM. arXiv preprint arXiv:2503.10009 (2025)

A Generated Constraint Gallery

We give two generated plugins verbatim (apart from removed comments), each posting only `model.Add(...)` relations over the shared `assign` variables of Section 3.1: the worked example of Section 3.4 (Listing 1.2) and a second, aggregation-style constraint (test 117).

Listing 1.2. Generated plugin for "junior staff cannot do surgical treatments" (test 39, the worked example of Section 3.4).

```
def add_constraint(model, assign, needs_by_id, slots_by_id,
    context):
    junior = {sid for sid, s in context["staff"].items()
              if str(s.get("seniority_level","")).lower() ==
                  "junior"}
    surgical = {tid for tid, t in context["treatments"].
        items()
                if str(t.get("treatment_category","")).lower
                    () == "surgical"}
    for n in context["need_ids"]:
        for s in context["compatible_slots"].get(n, []):
            slot = slots_by_id.get(s)
            if slot and slot["staff_id"] in junior and slot[
                "treatment_id"] in surgical:
                model.Add(assign[(n, s)] == 0)
```

Aggregation (test 117): "each building should handle at most 8 appointments per day." The plugin groups assignment variables by the `(building, day)` key (an aggregation the base model never computes) and posts one Boolean-linear at-most-8 inequality per group.

```
def add_constraint(model, assign, needs_by_id, slots_by_id,
    context):
    from collections import defaultdict
    rooms = context["rooms"]
    by_bd = defaultdict(list)
    for n in context["need_ids"]:
```

```
        for s in context["compatible_slots"].get(n, []):
            slot = slots_by_id.get(s)
            room = rooms.get(slot["room_id"]) if slot else
                None
            if room and (n, s) in assign:
                by_bd[(room["building"], slot["day_of_week"
                    ])].append(assign[(n, s)])
    for vars_list in by_bd.values():
        model.Add(sum(vars_list) <= 8)
```

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Reasoning about University Study Regulations Module-based Student Constraints and Preferences in Answer Set Programming

Henry Otunuya[0000−0002−1012−7887] and Sebastian Schellhorn[0000−0003−4928−2515]

University of Potsdam, Germany
henry.chuks.otunuya@uni-potsdam.de
sebastian.schellhorn@uni-potsdam.de

Abstract. University study regulations are legal documents which usually induce an enormous amount of valid study plans from which students may choose. And with several other works dealing with courses and/or examinations and not necessarily the regulations behind them, this work is part of an attempt to shift perspectives. Indeed, the combinatorial solution space induced by university study regulations is not at all reasonable for students to explore manually, let alone check for the satisfaction of certain desirable characteristics. Hence, building upon our previous work, and as part of the effort in the digitalization of higher educational institutions, we hereby present four hard and soft constraints which students are able to apply to find study plans which suits their needs.We motivate and formalize the four constraints to reflect needs on the number of ECTS per semester, occurrences of modules in a specific semester range, the number of modules for a particular semester and the number of total semesters needed to finish studies. We use Answer Set Programming (ASP) to model these constraints. An ASP-driven user interface prototypes our implementation allowing for user interactions with the constraints and solution space exploration according to individual demands.

Keywords: Study regulations · Answer set programming · Study plan.

1 Introduction

With university study regulations[1], higher educational institutions are able to offer and regulate the modules and examinations required by students to earn a degree. In such offers, university study regulations are mostly open when it comes to what modules students can take and in what semester.

[1] Also known as *university study and examinations regulations*, in universities like in TU Berlin [16] and Heinrich-Heine-Universität Düsseldorf [7].

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

This openness creates both opportunities and challenges [9, 8]. The opportunities come from the freedom to choose whatever module a student finds interesting, when to take such module and what other module to combine with it. On the otherhand there are challenges, some of which include how to keep track that the general study regulations constraints are satisfied - such as total ECTS credit, dependency between selected modules and credit requirements before thesis. Other challenges deal with how to ensure that a given credits bound for certain semesters are accomplished, that certain modules are part (or not part) of a study plan or that a certain optimization is performed. Admittedly, the prominence of the different challenges become more evident with increasing number of modules.

Furthermore, some of those challenges were pointed out by students interviewed within the *CAVAS+* project[2] [14], which this work is part of, further drawing our attention to them within the university environment.

In [6], we presented a set-based mathematical model of a study regulations problem from the University of Potsdam in Germany. There, we formally defined some study regulations constraints using the M.Sc. Cognitive Systems study program as our test-bed. With that we were able to ensure that some general rules in study regulations are checked in an automated fashion. Answer Set Programming (ASP) [10], a declarative combinatorial problem solving approach, was used to encode the problem. This work simply builds on that foundational paper.

From literature, it was observed that many works deal with the scheduling of courses, and not the undelying regulations behind the offered courses [11, 15, 12]. We argue that it is crucial to have a connection to actual university study regulations as this aligns such offerings with the legal goals of the higher educational institution involved.

This work and our foundational paper is part of an attempt to fill that gap. Our goal is to present some student-related hard and soft constraints which can be introduced into the solution space of study plans, thereby addressing some challenges. We add four constraints to reflect needs on the number of ECTS per semester, occurrences of modules in a specific semester range, the number of modules for a particular semester and the number of total semesters needed to finish studies. An ASP-driven user interface prototypes our implementation allowing for user interactions with the constraints and solution space exploration according to individual demands.

The paper is structured as follows - in Section 2 we present the hard constraints followed by the associated soft constraints in a mathematically formal way, with all the soft constraints sharing a common optimization function. An introduction to ASP, the ASP encodings of the constraints and our user interface are discussed in Section 3. In Section 4 we provide some related works, and then conclude with some future works in Section 5.

[2] `https://www.uni-potsdam.de/en/cavas-plus-en/index`(last accessed: June 2026).

2 Formalization of hard constraints and preferences

With hard constraints students are able to specify the required characterics of any potential study plan. These are constraints which must be satisfied. On the other hand, when hard constraints are not satisfiable at all, there is the possibility of switching to soft constraints, which encode the preferences. The first and the last constraints presented here are relatable to questions asked by interviewed students within the *CAVAS+* project [14], while the other two were inspired from the desire of students to have individualized study plans reported in literature [9, 13].

Since students are possibly not always interested in solutions which strictly satisfy given constraints and sometimes it may not even be possible to find a solution which satisfies given hard constraints or their combinations, preferences come into play.

To this end, we define our objective function $dtb : \mathbb{N}^3 \to \mathbb{N}$ which is the distance from the value x to some lower bound l and upper bound u, written as $dtb(x, l, u)$, as:

$$dtb(x,l,u) = \begin{cases} l - x & \text{if} \quad x < l \\ x - u & \text{if} \quad x > u \\ 0 & \text{otherwise} \end{cases}$$

With *dtb* we are able to get a value close to l and u, and then optimize such values.

We define our soft constraints as so-called *preference relations*. A preference relation is a preorder over possible study plans. That is, $\preceq \subseteq (S_i)_{i=1}^n \times (S'_i)_{i=1}^k$ for all $\preceq \in \{\preceq_{sc}, \preceq_{ms}, \preceq_{sm}, \preceq_{sb}\}$[3], where $(S_i)_{i=1}^n$ and $(S'_i)_{i=1}^k$ are possible study plans from some *basic study regulation problem* $\mathcal{B}$[4].

For any two possible study plans $(S_i)_{i=1}^n$ and $(S'_i)_{i=1}^k$, $(S_i)_{i=1}^n$ is preferred or equal to $(S'_i)_{i=1}^k$ with respect to function *dtb* on some bound, iff $(S_i)_{i=1}^n \preceq (S'_i)_{i=1}^k$, for some given preference relation $\preceq \in \{\preceq_{sc}, \preceq_{ms}, \preceq_{sm}, \preceq_{sb}\}$.

With each of the preference relations, except $\preceq_{sb}$, the result of the summed up *dtb* values in the first study plan $(S_i)_{i=1}^n$ must be less than or equal to the result of the summed up *dtb* values in the second study plan $(S'_i)_{i=1}^k$. For $\preceq_{sb}$, no aggregation is performed, only a simple check.

In the following subsections, we present hard constraints and preferences which relate to modules in our study regulations.

[3] These are our preference relations, defined in the following subsections.

[4] Notations not defined here, such as $\mathcal{B}, (S_i)_{i=1}^n$, come for our foundational work. For quick reference we put them all in Appendix B.

2.1 Semester-to-ECTS credits bound

Each module has an ECTS value representing the workload needed in accomplishing the module. Functions $scl, scu : \{1, 2, \ldots, n\} \to \mathbb{N}$ assign semesters of length $n \in \mathbb{N}$ to lower and upper bounds credits respectively[5].

In order to bound the ECTS workload in each semester, we employ the above functions. Constraint $R_{sc} \subseteq 2^{(2^M)^n}$ ensures that the semesters are within their ECTS bounds:

$$R_{sc} = \{\{(S_i)_{i=1}^n \subseteq M^n \mid scl(i) \leq \textstyle\sum_{m \in S_i} c(m) \leq scu(i) \text{ for all } 1 \leq i \leq n\}\}$$

For example, let us consider a hypothetical student Paul, who wants to take between 30 to 42 ECTS credits in the second semester, nothing in the third and exactly 30 ECTS credits in the fourth. That is:

$$scl = \{2 \mapsto 30, 3 \mapsto 0, 4 \mapsto 30\}$$
$$scu = \{2 \mapsto 42, 3 \mapsto 0, 4 \mapsto 30\}$$

The preceding formalization is encapsulated by the tuple (B, scl, scu, R_{sc}), which represents a *study regulation problem with semester-to-ECTS credits bound*. For the preference, we introduce a shared framework which will be used for all the preferences presented in this paper.

Preference on semester-to-ECTS credits bound Given our *scl* and *scu* bound, we define the preference relation $\preceq_{sc}$ on semester-to-ECTS credits bound wrt possible study plans $(S_i)_{i=1}^n$ and $(S'_i)_{i=1}^k$ by $(S_i)_{i=1}^n \preceq_{sc} (S'_i)_{i=1}^k$ iff:

$$\sum_{i=1}^{n}(dtb(\sum_{m \in S_i} c(m), scl(i), scu(i))) \leq \sum_{i=1}^{k}(dtb(\sum_{m \in S'_i} c(m), scl(i), scu(i)))$$

The $\preceq_{sc}$ preference relation prefers study plans with total ECTS credit of all the given semester mappings closest to all associated scl, scu range values.

2.2 Module-to-semester bound

Furthermore, a student may want to assign modules to specific semesters. In order to require modules $M_I \subseteq M$ to be available within a certain semester bound, functions $msl, msu : M_I \to \{1, 2, \ldots, n\}$ assign each $m \in M_I$ to a lower bound and to an upper bound of semester of length $n \in \mathbb{N}$, respectively[6]. We

[5] If no bounds are given for any semester (respectively undefined), then *scl* and *scu* take 0 and ∞ by default, respectively.

[6] If no bounds are given for any module (respectively undefined), then *mcl* and *mcu* take 1 and n by default, respectively.

introduce constraint $R_{ms} \subseteq 2^{(2^M)^n}$ to ensure that modules in M_I always belong to the resulting study plans and stay within their assigned bounds:

$$R_{ms} = \{\{(S_i)_{i=1}^n \subseteq M^n \mid M_I \subseteq \overline{(S_i)_{i=1}^n}\},$$
$$\{(S_i)_{i=1}^n \subseteq M^n \mid m \in M_I \cap S_i, msl(m) \leq i \leq msu(m) \text{ for all } 1 \leq i \leq n\}\}$$

With this Paul wants to state that module $am_{1,1}$ must be accomplished in the first three semesters and module pm_2 can occur only in the second semester. That is:

$$M_I = \{am_{1,1}, pm_2\}$$
$$msl = \{am_{1,1} \mapsto 1, pm_2 \mapsto 2\}$$
$$msu = \{am_{1,1} \mapsto 3, pm_2 \mapsto 2\}$$

The preceding formalization is encapsulated by the tuple $(B, M_I, msl, msu, R_{ms})$, which represents a *study regulation problem with module-to-semester bound*.

Preference on module-to-semester bound Given our msl and msu bound, we define the preference relation $\preceq_{ms}$ on module-to-semester bound wrt possible study plans $(S_i)_{i=1}^n$ and $(S'_i)_{i=1}^k$ by $(S_i)_{i=1}^n \preceq_{ms} (S'_i)_{i=1}^k$ iff:

$$\sum_{\substack{i=1 \\ m \in M_I \cap S_i}}^{n} (dtb(i, msl(m), msu(m))) \leq \sum_{\substack{i=1 \\ m \in M_I \cap S'_i}}^{k} (dtb(i, msl(m), msu(m)))$$

The $\preceq_{ms}$ preference relation prefers study plans with modules of all the given modules mappings closest to all associated msl, msu range values.

2.3 Semester-to-number of modules bound

Sometimes students may want to ensure that a certain number of modules are available in each semester. We generalize this constraint with the number of modules bound. This will allow a student to set a lower and an upper bound. The lower and upper bounds for a module are assigned by functions $sml, smu : \{1, 2, \ldots, n\} \to \mathbb{N}$ respectively[7], for semesters of length $n \in \mathbb{N}$.

Constraint $R_{sm} \subseteq 2^{(2^M)^n}$ ensures that for each semester with a set bound, such bound is respected:

$$R_{sm} = \{\{(S_i)_{i=1}^n \subseteq M^n \mid sml(i) \leq |S_i| \leq smu(i) \text{ for all } 1 \leq i \leq n\}\}$$

For example, Paul wants to take at least five modules in the first semester and limits his second semester by at most four modules. That is:

$$sml = \{1 \mapsto 5\}$$
$$smu = \{2 \mapsto 4\}$$

[7] If no bounds are given for any semester (respectively undefined), then sml and smu take 0 and ∞ by default, respectively.

The preceding formalization is encapsulated by the tuple (B, sml, smu, R_{sm}), which represents a *study regulation problem with semester-to-number of modules bound*.

Preference on semester-to-number of modules bound Given our sml and smu bound, we define the preference relation $\preceq_{sm}$ on semester-to-number of modules bound wrt possible study plans $(S_i)_{i=1}^n$ and $(S'_i)_{i=1}^k$ by $(S_i)_{i=1}^n \preceq_{sm} (S'_i)_{i=1}^k$ iff:

$$\sum_{i=1}^{n}(dtb(|S_i|, sml(i), smu(i))) \leq \sum_{i=1}^{k}(dtb(|S'_i|, sml(i), smu(i)))$$

The $\preceq_{sm}$ preference relation prefers study plans with semesters of all the given semester mappings closest to all associated sml, smu range values.

2.4 Number of semesters bound

Another need of students often is, to define the number of semesters they want to study in. Formally, $sl, su \in \mathbb{N}$ respectively holds the lower and the upper bound for any study plan. Constraint $R_s \subseteq 2^{(2^M)^n}$ ensures that all resulting study plans are within the given semester bounds:

$$R_s = \{\{(S_i)_{i=1}^n \subseteq M^n \mid s_l \leq n \leq s_u\}\}$$

For example, Paul wants to finish is study within 4 or 5 semesters. That is:

$$sl = 4$$
$$su = 5$$

The preceding formalization is encapsulated by the tuple (B, sl, su, R_s), which represents a *study regulation problem with number of semesters bound*.

Preference on number of semesters bound Given our sl and su bound, we define the preference relation $\preceq_{sb}$ on number of semesters bound wrt possible study plans $(S_i)_{i=1}^n$ and $(S'_i)_{i=1}^k$ by $(S_i)_{i=1}^n \preceq_{sb} (S'_i)_{i=1}^k$ iff:

$$dtb(n, sl, su) \leq dtb(k, sl, su)$$

The $\preceq_{sb}$ preference relation prefers study plans with semester length closest to sl, su range values.

Each combination of hard constraints in the preceding subsections interact with the general constraints in $\mathcal{B}$ in a similar fashion. For example, if we consider sl, su and R_s in addition to $\mathcal{B}$, the resulting study plan $(S_i)_{i=1}^n$ respects $(S_i)_{i=1}^n \in \bigcap_{r \in R_s \cup R_g \cup R_t} r$.

In the case of Paul and applying all his hard constraints,

$$(S_i)_{i=1}^4 = (\{bm_1, bm_3, am_{1,1}, am_{1,2}, am_{3,1}, pm_1\}, \{bm_2, im, am_{3,2}, pm_2\}, \{\}, \{msc\})$$

would be a possible solution, with total ECTS credits of 48, 42, 0 and 30 for semesters 1 to 4, respectively.

Each preference means something different and together allows a student to have some desired study plan(s).

3 ASP Encoding and User Interface

ASP is a logic-based paradigm suitable for solving combinatorial problems. An ASP program is a set of rules of the form

$$\texttt{a}_0 \texttt{ :- a}_1\texttt{,...,a}_m\texttt{,not a}_{m+1}\texttt{,...,not a}_n$$

where each $\texttt{a}_i$ is an atom of form $\texttt{p(t}_1\texttt{,...,t}_k\texttt{)}$ and all $\texttt{t}_i$s are terms, composed of function symbols and variables. For $0 \leq m \leq n$, atom $\texttt{a}_0$ is often called the head atom, while $\texttt{a}_1$ to $\texttt{a}_m$ and $\texttt{not a}_{m+1}$ to $\texttt{not a}_n$ are also referred to as positive and negative body literals, respectively. An expression is said to be ground, if it contains no variables.

A rule is called a fact if $m = n = 0$, and an integrity constraint if the head, $\texttt{a}_0$, is empty. Intuitively, the rule says that if $\texttt{a}_1$ to $\texttt{a}_m$ are true and $\texttt{a}_{m+1}$ to $\texttt{a}_n$ are false, then $\texttt{a}_0$ must be true.

Semantically, an ASP program induces a set of stable models, being distinguished models of the program determined by the stable models semantics [5].

For an in-depth coverage on ASP, we direct the interested reader to [10, 4].

3.1 Encoding

Here, we talk about the ASP encoding of the constraints presented in the previous section - for brevity we only focus on the first hard and soft constraints. The other encodings are in Appendix A. Note that the ASP encoding which generates the study plans was already presented in our previous paper, on top of which the following encoding operates. Additionally, as in that paper, our encoding was solved with the ASP system *clingo*[8].

For the hard constraints, we re-use the `map(F,E,V)` predicate which defines functions. That is, every function defined above is represented by `map(F,E,V)` which says that function `F` applied to element `E` gives value `V`. For example, with the *scl* and *scu* functions of the *semester-to-ECTS credits bound*, we have them as `map(scl,I,L)` and `map(scu,I,U)` respectively, with `I`, `L` and `U` being variables.

The complete encoding for the semester-to-ECTS credits bound constraint is given in Listing 1.1.

As a recap, `in(_,s(I))` holds modules which satisfy both the `map(scl,I,L)` and `map(scu,I,U)` generated from user interaction on the user interface, lines 2-4 show the hard constraints. There we say, if there is *semester-to-ECTS credits bound* constraint set for semester `I`, then the sum of the credits in that semester must be within the `L,U` bound.

[8] Version 5.4.0 - `https://potassco.org/clingo/` (last accessed: June 2026)

```
1  % hard constraint
2  :- in(_,s(I)), map(scl,I,L), map(scu,I,U),
3      not L #sum{ V,M : in(M,s(I)), map(c,M,V) } U,
4      not checked_pref(map(scl,I,L)), not checked_pref(map(scu,I,U)).

6  % soft constraint
7  map(dtb,I,D) :- in(_,s(I)), map(scl,I,L), map(scu,I,U),
8                  X = #sum{ V,M : in(M,s(I)), map(c,M,V) }, X < L,
9                  D = L - X.
10 map(dtb,I,D) :- in(_,s(I)), map(scl,I,L), map(scu,I,U),
11                 X = #sum{ V,M : in(M,s(I)), map(c,M,V) }, X > U,
12                 D = X - U.

14 #minimize{ D,I: map(dtb,I,D), map(scl,I,L), checked_pref(map(scl,I,L)),
15                                map(scu,I,U), checked_pref(map(scu,I,U)) }.
```

Listing 1.1. Hard and soft constraint of *semester-to-ECTS credits bound* constraint in `semester_to_ects.lp`.

Atom `map(dtb,I,D)` holds the *dtb* function value `D` for semester `I`. This predicate is used in lines 14-15 to encode the optimization of the associated constraint.

The `checked_pref/1` predicate (with arity 1) is connected to the user interface and will be explained in the next subsection. The other constraints in Appendix A have a similar structure with different ways for generating the *dtb*-related atoms. The encodings of the other constraints can be found in their associated files in the repository[9].

3.2 User Interface

In this section, we present our ASP-driven user interface made with the *clinguin* system[10]. We only discuss the newly introduced ui-related atoms here and for the basis refer the interested reader to the foundational paper [6].

Additionally, for an in-depth look into the *clinguin* system the interested reader is referred to [2]. Our implementation has a so-called *custom backend*[11] which defines Python functions for handling the adding and removing, the conversion into preferences and the off/on switching of constraints[12].

[9] `https://github.com/krr-up/study-regulations-modules-constraints-encodings/tree/main/encodings` (last accessed: June 2026)

[10] Version 2.8.4 - `https://clinguin.readthedocs.io/en/latest/` (last accessed: June 2026)

[11] `https://github.com/krr-up/study-regulations-modules-constraints-encodings/blob/main/encodings/user_interface/backend.py` (last accessed: June 2026)

[12] Functions `add_constraint`, `remove_constraint` and `make_preference` are implemented in `backend.py`.

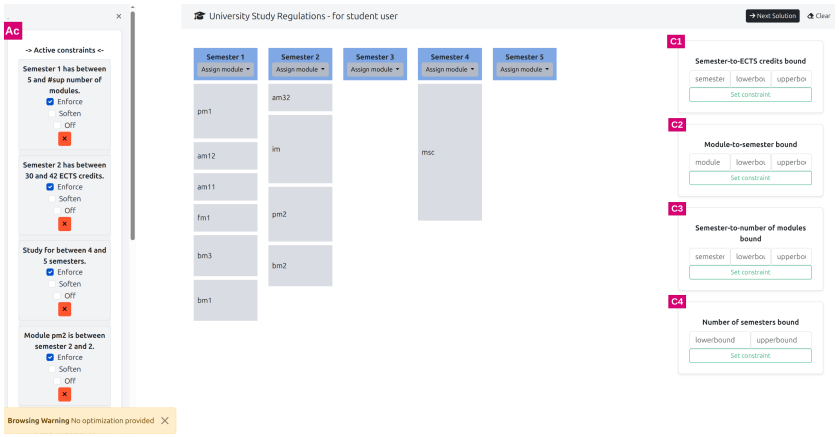


Fig. 1. User interface with constraints

The `checked_pref/1` predicate is generated when the user checks the *Soften* checkbox and the corresponding constraint passed as the value, and removed when the checkbox is unchecked.

Figure 1 shows the screenshot our *clinguin* implementation[13]. Labels **C1** (for *semester-to-ECTS credits bound* constraint), **C2** (for *module-to-semester bound* constraint), **C3** (for *semester-to-number of modules bound* constraint), **C4** (for *number of semesters bound* constraint), and **Ac** were added to the screenshot to aid the following explanation.

With study plans shown in the middle of the user interface, labels **C1** to **C4** show HTML cards for the four constraints. **C1** to **C3** have three input fields and **C4** two fields. With **C1**, a student is able to enter a semester with the lowerbound and upperbound ECTS credits values. Clicking on *Set constraint* introduces and activates a constraint and limits the solution space accordingly.

With **C2**, a student is able to enter a module and set the lower and upper bound of its semester values. With **C3**, a student is able to enter a semester with the lower and upper bound number of modules for that semester. And lastly with **C4**, the lower and upper bound semester values can be set.

The **Ac** sidebar shows the activated constraints. Each of the activated constraints share the same options to refine them. First, the constraint is reflected by an English sentence, followed by the *Enforce*, *Soften* and *Off* checkboxes. The first two checkboxes behave like HTML radio-buttons, that is, a constraint is either in *Enforce* or in *Soften*, but not in both. For instance, the first activated constraint belongs to **C3** and meets Paul's desire to accomplish 5 to #sup[14] number of modules in the first semester.

[13] Note that we used five semesters simply for the sake of demonstrating the working of our preferences, as explained below.

[14] #sup in *clingo* is supremum which *stands for* ∞.

Per default, *Enforce* is checked when a constraint is activated. *Soften*, when checked, turns the associated hard constraint into a soft constraint. A soft constraint minimizes the *dtb* function value shared among the constraints **C1** to **C4**. This objective function essentially holds the deviation from a given lower and upper bound.

Checkbox *Off*, when checked, deactivates the associated constraint. Thus, a user is able to toggle a particular constraint on and off.

The button with ×, removes a constraint and relaxes the solution space. Similarly, the *Clear all constraints* button removes all active constraints from the user interface and the underlying encoding.

The full user interface encoding can be found in the *user_interface* directory of our repository[15].

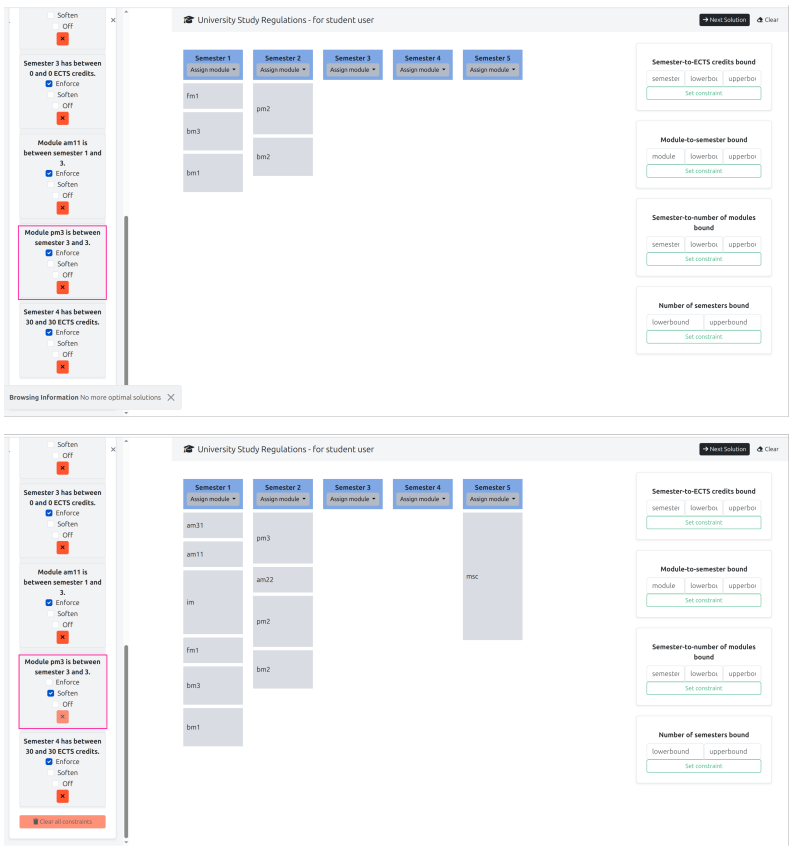


Fig. 2. User interface with violating constraint

[15] `https://github.com/krr-up/study-regulations-modules-constraints-encodings/blob/main/encodings/user_interface` (last accessed: June 2026)

For the sake of illustration, if Paul was to require that the module pm_3 be placed in the third semester, there would be no study plan since it violates his above constraint which says "no module should be in the third semester".

In Figure 2, we see that this is addressed by an info message at the bottom of the **Ac** sidebar which says "Browsing Information No more optimal solutions". However, softening the constraint gives a solution where module pm_3 is close to the third semester.

We observed a solving latency of less than one second with the adding and reasoning with the different constraints[16]. Due to this and since our problem tackles an NP-hard problem with a reasonable small number of modules and semesters, we are not yet concerned with performance, but would evaluate that as part of a usability evaluation in the future.

4 Related Works

Most of the works we encountered in literature are not about university study regulations, rather they deal with actual courses or process-mined event logs which operate on some possible university regulations [3, 17, 1].

However, [9] presented a university study planning tool for students which was based on examination regulations. They provided the user interface for study planning, but no modeling or reasoning engine for the study plans given. [11] presented a university course recommendation framework named *Hybrid Course Recommendation*. It relied on the following scores from a student - *interest-based* score, *timing-based* score and *grade-based* score for making student recommendations using historic data. Though, the solver used was not mentioned in the paper.

Along the same line, [12] presented an interactive university course recommendation system named *CourseQ*. The authors employed, among other things, topic-modeling to recommend courses related to different topic areas using the *Latent Dirichlet Allocation (LDA)* generative probabilistic model. They also reasoned with student interests.

This work is distinguished from those in that it containts student-related module constraints mathematically formalized, to the best of our knowledge, for the first time. This is also the first time *clinguin* is used to implement the presented university study regulations constraint concepts, having the ability to switch the constraints on and off (and back), and from hard to soft (and back).

5 Conclusion and Future Works

We have introduced four student-related hard and soft constraints formalizations, modeling and implementation in Answer Set Programming (ASP). Our

[16] The execution was carried out on a Linux machine equipped with Intel® Core™ i7-6700 × 8 processor and 32 GiB RAM.

approach does not only capture the core of the problem which are the constraints, but also the design of a user interface, thereby showing the versatility of our ASP approach.

The constraints are fed in as ASP facts which are then reasoned within the encodings, and displayed in the user interface. The *clinguin* user interface, which serves as a prototype, allow a user to interact with the constraints switching between hard and soft constraints and resulting in personalized study plans.

For future work, we would like to explore the possibility of reasoning with other kinds of constraints like *liked/disliked modules* and *module keywords*, *shortest study plan* and *dependent modules distance*. Also, we would like to explore how to give *explanations* when no study plan is found. And lastly, perform some usability evaluation with students and explore reasoning from the perspective of *study regulations designers*, for the analysis of study regulations and generation of exemplary study plans.

Acknowledgements. This work was partially funded by the German Federal Ministry of Research, Technology and Space under contract number 16DHBKI024. We also thank Torsten Schaub for supervising the initial developement of this work, Javier Romero for guiding with the preference formalization at the initial stage and Mariia Merzliakova whose *clinguin* study regulations ground work within *CAVAS+* was adopted and extended.

Disclosure of Interests. The authors declare no competing interests.

References

1. C. Behuet et al. “A Rule Language and Feature Model for Educational Timetabling”. In: *14th International Conference on the Practice and Theory of Automated Timetabling*. 2024.
2. A. Beiser, S. Hahn, and T. Schaub. “ASP-driven User-interaction with Clinguin”. In: 2024. URL: https://dx.doi.org/10.4204/EPTCS.416.19.
3. A. Bogarin, R. Cerezo, and C. Romero. “A survey on educational process mining”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.1 (2018), e1230.
4. M. Gebser et al. *Answer set solving in practice*. Cham: Springer International Publishing, 2013. ISBN: 978-3-031-01561-8. URL: https://doi.org/10.1007/978-3-031-01561-8_1.
5. M. Gelfond and V. Lifschitz. “Logic Programs with Classical Negation”. In: pp. 579–597.
6. S. Hahn et al. “Reasoning About Study Regulations in Answer Set Programming”. In: *Theory and Practice of Logic Programming* 24.4 (2024), pp. 790–804. URL: https://doi.org/10.48550/arXiv.2408.04528.
7. HHU. *Studien- und Prüfungsordnungen*. https://www.hhu.de/en/studies/organising-your-studies/examinations/study-and-exam-regulations-german-only. [Accessed 14-12-2025].

8. T. Hirmer, J. Etschmann, and A. Henrich. “Requirements and prototypical implementation of a study planning assistant in CS programs”. In: *2022 International Symposium on Educational Technology (ISET)*. IEEE. 2022, pp. 281–285.
9. S. Judel et al. “Supporting individualized study paths using an interactive study planning tool”. In: Gesellschaft für Informatik eV, 2023.
10. V. Lifschitz. *Answer set programming*. Vol. 3. Springer Heidelberg, 2019. ISBN: 978-3-030-24657-0. URL: https://doi.org/10.1007/978-3-030-24658-7.
11. B. Ma, Y. Taniguchi, and S. Konomi. “Course Recommendation for University Environments.” In: *International educational data mining society* (2020).
12. B. Ma et al. “Exploration and Explanation: An Interactive Course Recommendation System for University Environments.” In: *IUI Workshops*. 2021.
13. M. Ochs et al. “Design-focused development of a course recommender system for digital study planning”. In: *European Conference on Advances in Databases and Information Systems*. Springer. 2023, pp. 575–582.
14. H. Otunuya, S. Lindow, and M. Merzliakova. “Reasoning over Student Study Plans with Answer Set Programming”. In: *INFORMATIK 2025*. Bonn: Gesellschaft für Informatik e.V., 2025, p. 1143. DOI: 10.18420/inf2025_97.
15. A.L. Richards and R. Tsay. “An Optimal Slack-Based Course Scheduling Algorithm for Personalised Study Plans”. In: *Proceedings of the 2020 9th International Conference on Educational and Information Technology*. ICEIT 2020. Oxford, United Kingdom: Association for Computing Machinery, 2020, pp. 1–7. ISBN: 9781450375085. DOI: 10.1145/3383923.3383925. URL: https://doi.org/10.1145/3383923.3383925.
16. TU Berlin. *Study and examination regulations for all degree programs*. https://www.tu.berlin/en/pruefungen/examinations/study-and-examination-regulations/. [Accessed 14-12-2025].
17. M. Wagner et al. “A Combined Approach of Process Mining and Rule-based AI for Study Planning and Monitoring in Higher Education”. In: *Proceedings of the International Conference on Process Mining (ICPM’22): Process Mining Workshops*. Springer-Verlag, 2023, pp. 513–525.

Appendix

Appendix A

```
% -- mi set
in(M,mi) :- map(msl,M,_).
in(M,mi) :- map(msu,M,_).

% -- hard constraint
:- in(M,mi), not in(M,s(_)).

:- in(M,s(I)), map(msl,M,L), I < L, not checked_pref(map(msl,M,L)).
:- in(M,s(I)), map(msu,M,U), I > U, not checked_pref(map(msl,M,U)).

% -- soft : distance to bound L and U (dtb) -
map(dtb,M,D) :- in(M,s(X)), in(M,mi), map(msl,M,L), map(msu,M,U), X < L,
                D = L - X.
map(dtb,M,D) :- in(M,s(X)), in(M,mi), map(msl,M,L), map(msu,M,U), X > U,
                D = X - U.

#minimize{ D,M: map(dtb,M,D), map(msl,M,L), checked_pref(map(msl,M,L)),
                               map(msl,M,U), checked_pref(map(msl,M,U)) }.
```

Listing 1.2. Hard and soft constraint of *module-to-semester bound* constraint in `module_to_semester.lp`.

```
% hard constraint -
% if semester I is in a solution, then the total count of the modules in
% I must be between L and U.
:- in(_,s(I)), map(sml,I,L), map(smu,I,U), not L #count{M : in(M,s(I)) } U.

% soft constraint : distance to L and U bound (dtb) -
map(dtb,I,D) :- in(_,s(I)), map(sml,I,L), X = #count{ M: in(M,s(I)) }, X < L,
                D = L - X, not checked_pref(map(sml,I,L)).
map(dtb,I,D) :- in(_,s(I)), map(smu,I,U), X = #count{ M: in(M,s(I)) }, X > U,
                D = X - U, not checked_pref(map(smu,I,U)).

#minimize{ D,I: map(dtb,I,D), map(sml,I,L), checked_pref(map(sml,I,L)),
                              map(smu,I,U), checked_pref(map(smu,I,U)) }.
```

Listing 1.3. Hard and soft constraint of *semester-to-number of modules bound* constraint in `semester_to_number_of_modules.lp`.

```
% hard constraint
:- map(sl,L), n < L, not checked_pref(map(sl,L)).
:- map(su,U), n > U, not checked_pref(map(su,U)).

% soft constraint : distance to bound L and U (dtb) -
map(dtb,D) :- map(sl,L), map(su,U), n < L, D = L - n.
map(dtb,D) :- map(sl,L), map(su,U), n > U, D = n - U.

#minimize{ D: map(dtb,D), map(sl,L), checked_pref(map(sl,L)),
                          map(su,U), checked_pref(map(su,U)) }.
```

Listing 1.4. Hard and soft constraint of *number of semesters bound* constraint in `number_of_semesters.lp`.

Appendix B

A basic study regulation problem $\mathcal{B}$ is a tuple $(M, G, c, s, l, u, R_g, R_t)$, where

1. M is a set of modules,
2. $G \subseteq 2^M$ distinguishes certain groups of modules,
3. $c : M \to \mathbb{N}$ gives the credits of each module,
4. $s : M \to \{w, s, e\}$ assigns a regular semester to a module,
5. $l : G \to \mathbb{N}$ returns the lower-bound of the credits of a module group,
6. $u : G \to \mathbb{N}$ returns the upper-bound of the credits of a module group,
7. $R_g \subseteq 2^{(2^M)^n}$ is a set of global constraints expressing study regulations, and
8. $R_t \subseteq 2^{(2^M)^n}$ is a set of temporal constraints expressing study regulations.

Functions c and s give the credit points of a module and its semesters turnus, respectively. The elements l and u are partial functions delineating the number of credits obtained per module group; a specific number of credits is captured by an equal lower and upper bound.

The regulation constraints in R_g and R_t are represented extensionally: each constraint r in $R_g \cup R_t$ is represented by the sequences of sets of modules $r \subseteq (2^M)^n$ satisfying it. While the sets of constraints R_g and R_t share the same mathematical structure, they differ in purpose and are therefore separated for clarity's sake. R_t expresses temporal constraints over the sequence of semesters, while R_g does not make use of this sequential structure, and rather expresses global constraints over the entire set of modules.

A study plan for a basic study regulation is a finite sequence of sets of modules satisfying all regulation constraints. More precisely, a sequence $(S_i)_{i=1}^n$ of modules of length n such that $S_i \subseteq M$ for $1 \leq i \leq n$ is a *study plan* for $(M, G, c, s, l, u, R_g, R_t)$ if $(S_i)_{i=1}^n \in \bigcap_{r \in R_g \cup R_t} r$.

Configuration Support for Lexicographic Multi-Objective Scheduling: A Framework Applied to Electric Vehicle Charging

Laurenz Tomandl[1], Günther Raidl[1],
Tobias Rodemann[2], and Steffen Limmer[2]

[1] TU Wien, Vienna, Austria
[2] Honda Research Institute Europe GmbH, Offenbach/Main, Germany
{ltomandl|raidl}@ac.tuwien.ac.at,
{tobias.rodemann|steffen.limmer}@honda-ri.de

Abstract. Real-world scheduling problems frequently involve many, often conflicting objectives. Lexicographic optimization is a widely used approach for such many-objective problems, expressing the objectives' relative importance through a fixed priority order. However, as the number of objectives grows, configuring an effective order becomes challenging: optimizing certain objectives before others restricts the search space and the still obtainable values for remaining objectives in complex ways, and some objectives may dominate others or induce costly trade-offs. In this work, we present a data-driven framework for analyzing and supporting the configuration of lexicographic optimization in many-objective scheduling problems formulated as linear programs. We consider the space of all objective function permutations and compile an objective order graph that captures how the solution space and diverse quantitative measures evolve as objectives are fixed sequentially. Merging and pruning strategies recognize and remove symmetries, substantially reducing the graph. We demonstrate the framework on a smart electric vehicle charging scheduling problem with seven objectives deployed in a real-world setting. Results reveal strong asymmetries between objectives and show that only a small fraction of all possible objective orders lead to distinct outcomes. We further present an interactive tool that allows users to explore objective orderings and their consequences.

Keywords: Lexicographic optimization, many-objective scheduling, electric vehicle charging, decision support, objective ordering

1 Introduction

Scheduling problems in practice frequently involve multiple, often conflicting objectives that must be balanced simultaneously. In domains such as transportation, energy management, and timetabling, decision makers face the challenge of prioritizing objectives whose interactions are difficult to foresee. Computing the entire Pareto front is often computationally infeasible or unnecessary. A common alternative is to combine multiple objectives into a single objective, e.g.,

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

by means of a weighted sum, but this requires the careful selection of suitable weights, which is often difficult and unintuitive, and can easily produce misleading outcomes [12].

Lexicographic optimization provides a more intuitive alternative: objectives are optimized sequentially according to a fixed priority order, and once an objective has been optimized, its value is fixed by adding a constraint before proceeding to the next. Formally, let X be a feasible solution space and $f(x) = (f_1(x), \ldots, f_k(x))$, $x \in X$, a vector of objective functions. For $u, v \in \mathbb{R}^k$, define $u \prec v$ iff there exists an index $j = \min\{i : u_i \neq v_i\}$ such that $u_j < v_j$. The lexicographic optimization problem

$$\text{lexmin}_{x \in X} \ (f_1(x), \ldots, f_k(x))$$

asks for an $x^* \in X$ such that for any $x \in X$, $f(x^*) \prec f(x)$ or $f(x^*) = f(x)$ [5].

In many scheduling applications, such an ordering is easier to specify than numerical weights. However, when more than a few objectives are involved, configuring an effective order is far from trivial. Objectives considered early can drastically restrict the remaining feasible solution space, potentially rendering later objectives irrelevant or inducing severe trade-offs.

In this work, we propose a data-driven framework for analyzing objective orderings in lexicographic optimization of (mixed integer) linear programs (MILPs/LPs). We introduce an *objective order graph* (OOG), which compactly encodes restricted search spaces and achievable objective values for any partial objective order, and derive statistics characterizing objective interactions and order sensitivity. We demonstrate the framework on a smart electric vehicle (EV) charging scheduling problem with seven objectives, showing that only a small fraction of all orderings lead to distinct outcomes.

The remainder of this paper is structured as follows. Section 2 reviews related work. Section 3 introduces the OOG and its properties. The EV charging case study is described in Section 4, followed by analysis results in Section 5. Section 6 concludes the paper.

2 Related Work

Existing work in lexicographic optimization focuses mostly on algorithmic refinements or specific applications. Cococcioni et al. [4] embed lexicographic priorities into single-objective formulations using the Grossone methodology, while Lai et al. [10,11] study mixed Pareto-lexicographic problems. In metaheuristic optimization, Branke [1] and Gutiérrez et al. [2,3] use lexicographic orderings as preference information to guide search. For a broader perspective, we refer to the survey by Halffmann et al. [8] and to scalarization methods [14] as an alternative.

Closest to our work, Khorram et al. [9] perform a sensitivity analysis on objective priority in lexicographic multi-objective linear programs, showing how to efficiently re-solve the problem when the ordering changes. Their work focuses on adapting solutions to a given new ordering; in contrast, our framework

systematically analyzes and compares all orderings simultaneously to support configuration decisions. Related to this work is that of Eskelinen and Miettinen [7], who analyze local trade-offs in the neighborhood of a selected Pareto-optimal solution in scalarization-based multi-objective optimization, but they do not consider lexicographic orderings of objectives.

In industrial settings, lexicographic optimization is applied for its simplicity and interpretability, including electric vehicle charging [12] and production scheduling [13]. As a concrete test case, we use the hierarchical smart EV charging scheduler of Limmer et al. [12], deployed in a real-world charging management system [6,15].

3 The Objective Order Graph

Consider a many-objective (mixed integer) linear optimization problem solved via lexicographic optimization. We aim to capture how the search space and obtainable objective function values evolve across all meaningful permutations of objectives. To this end, we introduce a data structure we call the *objective order graph* (OOG). As a precursor, we first derive the *objective order tree* (OOT) before compressing and refining it into the OOG. In principle, the OOT is obtained by recursively considering all relevant permutations of the objectives. Each node in the tree represents a subspace of the original search space of the optimization problem, defined by constraints that fix a subset of objectives to the optimal values obtained at earlier stages of the lexicographic optimization process, with the root node corresponding to the original (unrestricted) feasible search space X of the optimization problem, i.e., before any objective has been fixed.

Each edge in the tree represents the act of optimizing a not yet considered objective within the search space of the source node and fixing the objective to the obtained optimal value in the target node. A complete OOT for k objectives thus has $k!$ leaves and $\sum_{i=0}^{k} k!/(k-i)!$ nodes in total (e.g., for $k = 7$: 5040 leaves and 13 700 nodes), since every prefix of every permutation corresponds to a distinct node. Consequently, each root-to-leaf path corresponds to one complete permutation of the objectives. The corresponding leaf represents the feasible search space remaining after all objectives have been fixed in that order.

We now formalize this in a more precise way. Let k be the number of objective functions, vector x be the decision variables of the MILP or LP by which the optimization problem is modeled, and $f_i(x)$ be the i-th objective function for $i = 1, \ldots, k$ in the natural (arbitrary) ordering. W.l.o.g., we assume that all objective functions are to be minimized. Let $a = (u, v)$ be an arc in the OOT from node u to node v. Let u be a node of the OOT. We associate the following data/functions with u:

- Let $d(u)$ be the depth of node u in the OOT, i.e., the number of arcs and therefore fixed objectives along the path from the root to u.
- Let $S(u)$ be the unordered set of indices of objectives along the path from the root to u, i.e., the index set of all already fixed objectives.

- Vector $f_{\min}(u) = (f_{\min}(u)_i)_{i=1,\dots,k}$, where $f_{\min}(u)_i$ is the smallest value attainable by the objective function f_i when it is minimized over the search space represented by node u.
- Vector $f_{\mathrm{ub}}(u) = (f_{\mathrm{ub}}(u)_i)_{i=1,\dots,k}$, where $f_{\mathrm{ub}}(u)_i$ is the largest value attainable by the objective function f_i when it is maximized independently over the search space represented by node u. These values serve as upper bounds and may not be attainable depending on the problem formulation, as meaningful upper bounds do not always exist.
- Vector $f_{\max}(u) = (f_{\max}(u)_i)_{i=1,\dots,k}$, where $f_{\max}(u)_i$ is the largest value that objective f_i actually takes across all final solutions reachable from node u, i.e., the component-wise maximum over the $f_{\min}$-vectors of all leaf nodes in the subtree rooted at u. Unlike $f_{\mathrm{ub}}(u)_i$, it reflects values actually occurring in solutions produced by the lexicographic process. This vector is computed bottom-up once the OOT has been fully constructed. For any leaf node u, all objectives are fixed and $f_{\min}(u)_i = f_{\max}(u)_i = f_{\mathrm{ub}}(u)_i$ for all $i = 1, \dots, k$. For inner nodes u, the bounds satisfy $f_{\min}(u)_i \leq f_{\max}(u)_i \leq f_{\mathrm{ub}}(u)_i$.

While the definitions above assume the complete OOT, this is not required in practice: large parts can be omitted by exploiting problem structure. Note that $f_{\min}(u)$, $f_{\max}(u)$, and $f_{\mathrm{ub}}(u)$ naturally extend to matrix representations when multiple problem instances are considered simultaneously, with each row corresponding to one instance.

Irrelevant Objectives and Nodes. We call an objective f_i *irrelevant* at node u when its fixation does not reduce the search space, i.e., $f_{\min}(u)_i = f_{\mathrm{ub}}(u)_i$. A node u is *irrelevant* if all unfixed objectives are irrelevant. Such nodes can be identified early during construction, avoiding unnecessary further exploration. Since the search space only shrinks along any root-to-leaf path, $f_{\min}(u)_i$ is non-decreasing and $f_{\mathrm{ub}}(u)_i$ non-increasing with depth. Consequently, once an objective is irrelevant at u, it remains irrelevant throughout the entire subtree rooted at u and cannot become relevant again deeper in the tree.

Problem Dependent Knowledge. Problem-specific or expert knowledge can be incorporated as precedence constraints on objectives, restricting which permutations are explored during OOT construction.

Subtree Merging. Different partial objective orders may lead to identical restricted search spaces. Two nodes u and v are *congruent* if they have fixed the same set of objectives ($S(u) = S(v)$) to the same optimal values ($f_{\min}(u)_i = f_{\min}(v)_i$ for all $i \in S(u)$), meaning that the search spaces represented by both nodes are identical and so are their entire subtrees. Note that $f_{\min}(u)$, $f_{\max}(u)$, and $f_{\mathrm{ub}}(u)$ depend only on this search space, not on the order in which the objectives in $S(u)$ were fixed. Merging congruent nodes by redirecting all incoming arcs of one to the other compresses the OOT into a directed acyclic graph, the *objective order graph* (OOG), which preserves all relevant information. In practice, congruent nodes are identified efficiently during construction via hashing on $S(\cdot)$ and the corresponding fixed objective values.

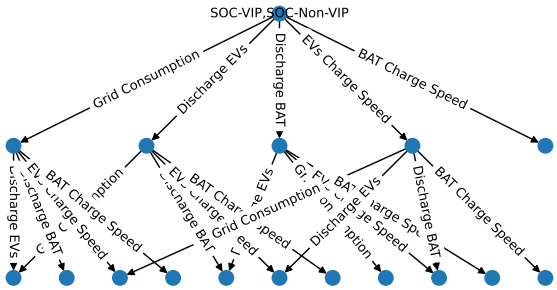


Fig. 1: Partial OOG rooted at the search space after fixing `SOC-VIP` and `SOC-Non-VIP`, shown to depth three. Nodes represent restricted search spaces; edges represent objective fixations. Multiple parents indicate that different orderings lead to the same search space (merging). The node reached by fixing `BAT-Charge-Speed` has no children because this fixation completely determines all remaining objectives.

Pruning of Order-Independent Postfixes. As an optional simplification, we remove *order-independent postfixes*: at a node u where $f_{\min}(u) = f_{\max}(u)$, the attainable minimum and maximum objective values coincide for all remaining objectives, meaning that fixing them in any order yields the same final solution. Consequently, all descendants of u can be removed, and u becomes a leaf. After this pruning step, the leaf nodes represent all decision-relevant prefixes of objective permutations, with $f_{\min}(u) = f_{\max}(u)$ directly giving the final objective values. This criterion differs from irrelevance: irrelevance is defined via the independent upper bound f_{ub}, whereas an order-independent postfix is defined via $f_{\max}$, the maximum actually attained across all solutions reachable from u, and $f_{\max}(u)_i \leq f_{\text{ub}}(u)_i$ in general. A node may therefore admit an order-independent postfix without any remaining objective being irrelevant. This step, however, discards structural information: in particular, details about how the search space is further restricted by the remaining objectives and how upper bounds (f_{ub}) evolve are lost. Since this information may be useful for diagnostic analyses, we treat the pruning as optional.

Figure 1 illustrates a partial OOG for our case study (see next Section 4), showing how different objective fixation orders can lead to the same restricted search space (merged nodes with multiple parents), and how a single objective fixation can completely determine all remaining objectives (leaf without children).

4 Case Study: Smart EV Charging Scheduling

As a case study, we consider the smart EV charging scheduler of Limmer et al. [12], deployed in a real-world charging management system at a company

building in Germany. The scheduler assigns charging power to vehicles and stationary batteries across discrete time slots, considering a photovoltaic system, two-way charging, and grid load constraints. The problem is formulated as a many-objective linear program with the following seven objectives to be minimized:

1. `SOC-VIP`: dissatisfaction of target state-of-charge (SOC) requirements of VIP EVs
2. `SOC-Non-VIP`: dissatisfaction of target SOC requirements of non-VIP EVs
3. `Grid-Consumption`: grid energy consumption
4. `Discharge-EV`: overall discharging of EV batteries
5. `Discharge-BAT`: overall discharging of stationary batteries
6. `EV-Charge-Speed`: negative objective prioritizing early charging and fully charging EVs
7. `BAT-Charge-Speed`: negative objective prioritizing early charging and fully charging batteries

Hard constraints ensure compliance with grid load limits, battery capacity bounds, and minimum SOC requirements. All decision variables are continuous. We treat the scheduler as a black box and refer to Limmer et al. [12] for the full model. The scheduler can be re-invoked with any (partial) objective order and reports the optimal value of each objective as it is fixed. We apply the scheduler to each subproblem associated with each OOG node and thereby compile the graph while imposing the precedence constraint that `SOC-VIP` must always precede `SOC-Non-VIP`.

5 Statistical Analysis

All experiments are conducted on three classes of smart EV charging scheduling instances with increasing problem size, containing 50, 100, and 200 electric vehicles, respectively. For each class, we generated 100 independent artificial instances that roughly mimic realistic scenarios, resulting in a total of 300 problem instances. These instances are available on our website[1]. We combine all 300 instances into a single aggregated OOG: the bound vectors $f_{\min}(u)$, $f_{\max}(u)$, and $f_{\text{ub}}(u)$ turn into matrices with one row per instance, and nodes are merged or pruned only if the respective relationship holds across all rows. As the framework targets offline configuration support, construction cost is non-critical. It is dominated by the underlying LP solve time and the number of explored permutations, which merging and pruning reduce substantially.

Based on this OOG, we derive descriptive statistics that quantify objective interactions, hierarchy sensitivity, and structural complexity. The goal is not to benchmark against alternative multi-objective approaches, but to characterize how objective orderings influence the solution space and final outcomes. We organize the results into node- and edge-level, objective-level, and graph statistics.

[1] `https://www.ac.tuwien.ac.at/files/resources/instances/HRI-charging_problem/instances_HRI_charging_problem_case_study.zip`

Table 1: Average bounding box size by objective and node depth across all test instances relative to the bounding box at depth 0.

	Depth					
Objective (i)	0 (root)	1	2	3	4	5
`SOC-VIP`	1.00	0.73	0.34	0.09	0.02	0.00
`SOC-Non-VIP`	1.00	0.63	0.31	0.16	0.06	0.00
`Grid-Consumption`	1.00	0.78	0.48	0.29	0.13	0.01
`Discharge-EV`	1.00	0.57	0.25	0.11	0.04	0.00
`Discharge-BAT`	1.00	0.60	0.22	0.09	0.04	0.00
`EV-Charge-Speed`	1.00	0.74	0.42	0.18	0.06	0.00
`BAT-Charge-Speed`	1.00	0.74	0.41	0.21	0.12	0.05

5.1 Node- and Edge-Related Statistics

At the level of individual nodes and arcs of the OOG, we analyze how the fixing of respective objectives restricts the remaining search space. For a node u, we consider the normalized bounding box size $\Delta(u)_i = (f_{\max}(u)_i - f_{\min}(u)_i)/(f_{\max}(r)_i - f_{\min}(r)_i)$ for $i = 1, \ldots, k$, where r denotes the root node. This measures the remaining variability of each objective relative to the full range at the root, yielding values in $[0, 1]$.

Table 1 shows average Δ-values over all nodes at each depth, across all instances. We observe a substantial decrease with increasing depth, confirming that earlier objectives are more impactful. The rate varies across objectives: some drop below 10% of their root value already at depth three, while others retain more flexibility.

To quantify the impact that fixing an individual objective along an arc $a = (u, v)$ has, we consider normalized arc-level reduction measures. The *relative range reduction* $\rho(a)_i$ captures the fraction of the remaining relative bounding box of an objective i, assuming $\Delta(u)_i > 0$. In addition, the *relative minimum increase* $\rho_{\min}(a)_i$ measures how strongly fixing the objective associated with arc a increases the minimum attainable value of objective i, i.e., actually worsens the best possible value of f_i.

$$\rho(a)_i = \frac{\Delta(u)_i - \Delta(v)_i}{\Delta(u)_i} \qquad \text{and} \qquad \rho_{\min}(a)_i = \frac{f_{\min}(v)_i - f_{\min}(u)_i}{f_{\max}(u)_i - f_{\min}(u)_i}$$

A value of $\rho(a)_i = 1$ means that f_i is completely fixed (dominated), while $\rho_{\min}(a)_i = 0$ means the attainable minimum is unaffected. We aggregate these into scalar indicators for arc a:

$$R(a) = \frac{1}{k - |S(v)|} \sum_{i \notin S(v)} \rho(a)_i, \qquad R_{\min}(a) = \frac{1}{k - |S(v)|} \sum_{i \notin S(v)} \rho_{\min}(a)_i$$

Tables 2 and 3 report average values of $R(a)$ and $R_{\min}(a)$ across depths. Values close to zero indicate little effect; values close to one indicate near-complete

Table 2: Average search space reduction indicator $R(a)$, averaged over all arcs a that fix the given objective at the given depth.

	Depth					
Objective of arc a	1	2	3	4	5	6
`SOC-VIP`	0.04	0.18	0.41	0.48	0.52	-
`SOC-Non-VIP`	-	0.72	0.58	0.61	0.64	1.00
`Grid-Consumption`	0.31	0.51	0.57	0.60	0.78	1.00
`Discharge-EV`	0.01	0.09	0.43	0.66	0.82	1.00
`Discharge-BAT`	0.23	0.18	0.15	0.15	0.23	1.00
`EV-Charge-Speed`	0.94	0.89	0.78	0.83	0.90	1.00
`BAT-Charge-Speed`	0.38	0.49	0.59	0.71	0.91	1.00

Table 3: Average minimum value increase $R_{\min}(a)$, averaged over all arcs a that fix the given objective at the given depth.

	Depth					
Objective of arc a	1	2	3	4	5	6
`SOC-VIP`	0.01	0.11	0.35	0.46	0.52	-
`SOC-Non-VIP`	-	0.35	0.27	0.36	0.45	1.00
`Grid-Consumption`	0.30	0.50	0.56	0.60	0.78	1.00
`Discharge-EV`	0.00	0.07	0.29	0.50	0.74	1.00
`Discharge-BAT`	0.04	0.06	0.09	0.09	0.19	1.00
`EV-Charge-Speed`	0.35	0.34	0.29	0.39	0.61	1.00
`BAT-Charge-Speed`	0.09	0.24	0.45	0.64	0.89	1.00

elimination of flexibility. A large gap between $R(a)$ and $R_{\min}(a)$ indicates a harmless restriction (search space shrinks but best values are preserved), while similar values of $R(a)$ and $R_{\min}(a)$ indicate a costly fixation.

The objective `EV-Charge-Speed` consistently attains large $R(a)$ values, meaning it almost completely collapses the remaining search space, yet its $R_{\min}(a)$ values are substantially lower, indicating that the remaining solutions are not necessarily bad for other objectives. In contrast, the `Grid-Consumption` objective shows comparable $R(a)$ and $R_{\min}(a)$ values, indicating that its fixation directly worsens the attainable values of remaining objectives. The empty entries for `SOC-Non-VIP` at depth one and `SOC-VIP` at depth six result from the precedence constraint and pruning, respectively.

5.2 Objective-Related Statistics

Figure 2 summarizes the average arc-level reduction measures $\rho(a)_i$ and $\rho_{\min}(a)_i$ over all arcs at any depth and all 300 instances (columns: fixed objective; rows: affected objective). The left plot reveals strong asymmetries: fixing `BAT-Charge-Speed` or `EV-Charge-Speed` leads to strong range reductions in other objectives. Most notably, `BAT-Charge-Speed` always dominates `Discharge-BAT` (complete range collapse), and `EV-Charge-Speed` exerts a similar influence on `BAT-Charge-Speed`. The right plot shows that ef-

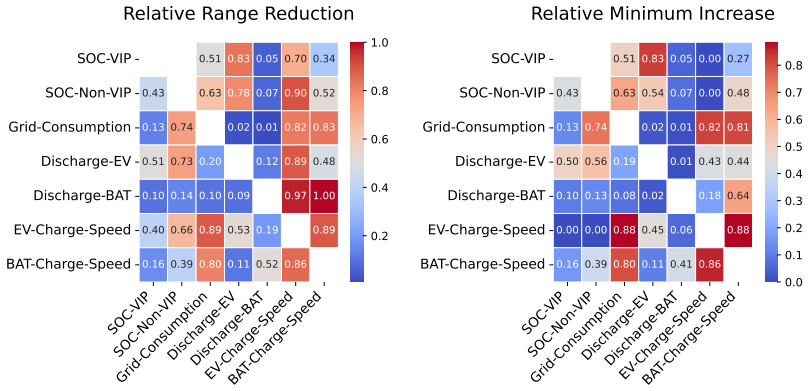


Fig. 2: Average *relative range reduction* and average *relative minimum increase* indicators when the objective in the respective column is optimized and fixed before the objective in the respective row.

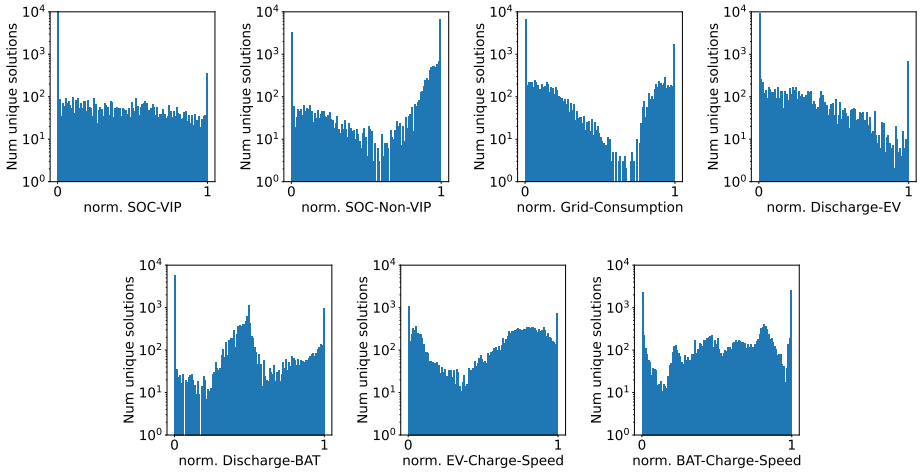


Fig. 3: Distributions of objective function values over all distinct final objective vectors in the OOGs, normalized per instance to $[0, 1]$, displayed on a logarithmic scale.

fects on ranges and on best attainable values can differ substantially: fixing `EV-Charge-Speed` does not worsen the minimum of `SOC-VIP` despite cutting its range considerably, whereas fixing `Grid-Consumption` reduces both range and minimum equally.

Figure 3 shows the distributions of the objective values of the final solutions over all objective orders, counting only distinct objective vectors, i.e., distinct leaf

Table 4: OOG remaining size after subtree merging and pruning, and number of distinct objective vectors.

Setting	Nodes after merging [%]	Leaves after merging [%]	Nodes after pruning [%]	Leaves after pruning [%]	Avg. (stddev.) distinct vectors
50 EVs	4.4	1.1	1.8	3.0	54.49 (11.49)
100 EVs	4.3	1.1	1.9	3.1	56.04 (12.79)
200 EVs	4.3	1.1	1.9	3.1	57.10 (11.67)
Multi	8.9	3.6	4.4	7.2	16763

$f_{\min}$ values after merging. Note that lexicographic optimization assigns a well-defined optimal objective vector to each order, even though the underlying LP optimum need not be a unique point in decision space. Values are normalized per instance using the root bounds $f_{\min}(r)$ and $f_{\max}(r)$, and the y-axis is logarithmic.

Most distributions exhibit a bimodal structure: solutions cluster near the best or worst attainable bounds, with few intermediate values, which is characteristic of lexicographic optimization where early objectives dominate. Notable exceptions are `Discharge-BAT`, `EV-Charge-Speed`, and `BAT-Charge-Speed`, which show broader spreads with substantial intermediate values.

5.3 Graph Statistics

Table 4 summarizes the OOG size after merging and pruning, relative to the full OOT (13 700 nodes, 5040 leaves). The first three rows average over the individual per-instance OOGs, while the "Multi" row is the single OOG combining all 300 instances. Subtree merging reduces the graph to about 4.4% of nodes, and postfix pruning further to 1.9%. On average, only about 56 distinct objective vectors exist per instance, a number remarkably stable across instance sizes. Note that the number of leaf nodes can increase after pruning, as removing a merged leaf with multiple predecessors exposes multiple new leaves at shallower depths. When combining all instances (row "Multi"), the OOG roughly doubles in size, indicating that orders prunable for some instances remain relevant for others.

5.4 Interactive Analysis Tool

To support practical configuration decisions, we developed a prototype tool that allows users to interactively explore the OOG by successively fixing objectives; cf. Figure 4. In this example, `Discharge-BAT` and `Grid-Consumption` have been fixed. The right panel shows histograms of objective value distributions for the current (blue) and preceding (red) search space, allowing immediate assessment of each decision's impact. The upper left panel shows the remaining graph structure, and the lower left panel reports $R(a)$ and $R_{\min}(a)$ values to guide the selection of the next objective.

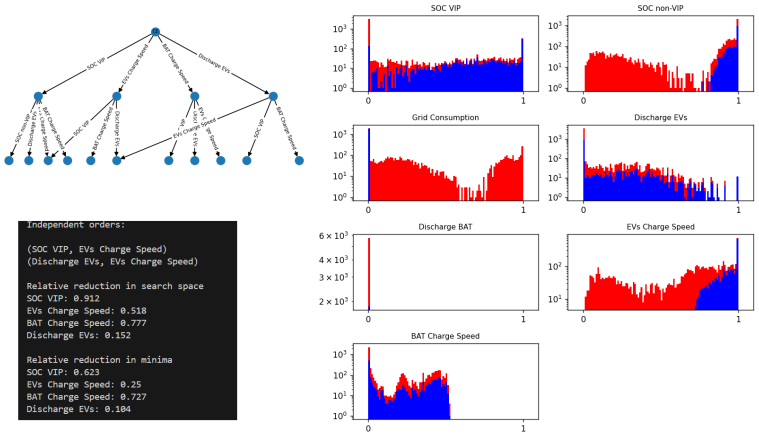


Fig. 4: User interface of prototypical interactive exploration tool.

6 Conclusion and Future Work

We introduced a data-driven framework for analyzing objective orders in lexicographic many-objective optimization. By constructing an *objective order graph* from the full space of objective permutations, merging congruent subtrees and pruning order-independent postfixes, we obtain a compact representation of all decision-relevant orderings. The derived *relative range reduction* and *relative minimum increase* indicators identify dominating objectives and costly trade-offs.

Applied to a real-world smart EV charging scheduling problem with seven objectives, the framework reveals substantial asymmetries between objectives that are difficult to anticipate a priori, highlighting the need for systematic configuration support.

Several directions for future work arise: (a) ranking objective orders by user-defined criteria; (b) relaxing the strict lexicographic order via a tolerance that permits controlled worsening of fixed objectives; (c) scaling to more objectives via more aggressive pruning or sampling; (d) comparing lexicographic and Pareto-based approaches on the same problem; (e) validating the framework on further scheduling and timetabling applications. Although our case study used a linear program, the approach applies to mixed-integer linear programs as well.

References

1. Branke, J.: Consideration of partial user preferences in evolutionary multiobjective optimization. In: Multiobjective Optimization, LNCS, vol. 5252, pp. 157–178. Springer (2008)
2. Castro-Gutiérrez, J., Landa-Silva, D., Moreno-Pérez, J.: Dynamic lexicographic approach for heuristic multi-objective optimization. In: Proceedings of the Work-

shop on Intelligent Metaheuristics for Logistic Planning (CAEPIA-TTIA 2009). pp. 153–163. Seville, Spain (2009)
3. Castro-Gutiérrez, J., Landa-Silva, D., Pérez, J.M.: Improved dynamic lexicographic ordering for multi-objective optimisation. In: Parallel Problem Solving from Nature, PPSN XI, LNCS, vol. 6239, pp. 31–40. Springer (2010)
4. Cococcioni, M., Pappalardo, M., Sergeyev, Y.D.: Towards lexicographic multi-objective linear programming using grossone methodology. AIP Conference Proceedings **1776**(1), 090040 (2016)
5. Ehrgott, M.: Multicriteria Optimization. Springer (2005)
6. Engel, J., Castellani, A., Wollstadt, P., Lanfermann, F., Schmitt, T., Schmitt, S., Fischer, L., Limmer, S., Luttropp, D., Jomrich, F., et al.: A real-world energy management dataset from a smart company building for optimization and machine learning. Scientific Data **12**(1), 1–19 (2025)
7. Eskelinen, P., Miettinen, K.: Trade-off analysis approach for interactive nonlinear multiobjective optimization. OR Spectrum **34**(3), 803–816 (2012)
8. Halffmann, P., Schäfer, L.E., Dächert, K., Klamroth, K., Ruzika, S.: Exact algorithms for multiobjective linear optimization problems with integer variables: A state of the art survey. Journal of Multi-Criteria Decision Analysis **29**(5-6), 341–363 (2022)
9. Khorram, E., Zarepisheh, M., Ghaznavi-ghosoni, B.: Sensitivity analysis on the priority of the objective functions in lexicographic multiple objective linear programs. European Journal of Operational Research **207**(3), 1162–1168 (2010)
10. Lai, L., Fiaschi, L., Cococcioni, M.: Solving mixed pareto-lexicographic multi-objective optimization problems: The case of priority chains. Swarm and Evolutionary Computation **55**, 100687 (2020)
11. Lai, L., Fiaschi, L., Cococcioni, M., Deb, K.: Pure and mixed lexicographic-paretian many-objective optimization: state of the art. Natural Computing **22**(2), 227–242 (2023)
12. Limmer, S., Attig, C., Rodemann, T.: A hierarchical controller as an alternative multi-objective EV charging manager. In: 16th International Conference on Power, Energy, and Electrical Engineering (CPEEE) (2026, in press, accepted)
13. Sawik, T.: Multi-objective master production scheduling in make-to-order manufacturing. International Journal of Production Research **45**(12), 2629–2653 (2007)
14. Sherali, H.D.: Equivalent weights for lexicographic multi-objective programs: Characterizations and computations. European Journal of Operational Research **11**(4), 367–379 (1982)
15. Stadie, M., T., R., Burger, A., Jomrich, F., Limmer, S., Rebhan, S., Saeki, H.: V2B vehicle to building charging manager. In: Proceedings of the EVTeC: 5th International Electric Vehicle Technology Conference. JSAE (2021)

MounouZinc: A MiniZinc-Based University Timetabling Tool Integrating AI Optimization and Web Automation

Tony Rouhayem[1], Youssef Bakouny[1], Laurent Drapeau[2], and Melhem El Helou[1]

[1] Faculty of Engineering and Architecture, Saint Joseph University of Beirut, Lebanon
[2] CNRS, IRD, INRAE, CEREGE, Aix-en-Provence, Aix Marseille University, Marseille, France

Abstract. University course timetabling is an NP-hard optimization problem defined by complex combinations of academic, pedagogical, and resource-related constraints. While recent advances in constraint programming and hybrid optimization have improved model expressiveness, institutions still lack integrated tools that connect optimization, data collection, recommendation, and student-support capabilities within a unified platform. This paper presents MounouZinc, a tool for university timetabling, designed as a modular built on MiniZinc. It integrates a PostgreSQL backend, a Python automation layer, and a web-based interface for collecting professor availability and timeslot preferences. The system integrates an AI-driven professor–course matching agent, which prioritizes professor assignments based on specialization and course descriptions, and a conversational chatbot that supports both new applicants (FAQ services) and current students (iAdvisor-style course selection and enrollment assistance). Experimental evaluations were conducted on synthetically generated datasets designed to reflect real institutional scenarios demonstrate that MounouZinc produces feasible, high-quality timetables that align with diverse academic policies while providing end-to-end decision-support functionalities rarely found in existing UCTP systems. MounouZinc advances beyond basic solving to offer an integrated decision-support platform.

Keywords: MiniZinc · Timetabling · Scheduling · Constraint Programming · Optimization · Artificial Intelligence · Chatbot · iAdvisor

1 Introduction

The University Course Timetabling Problem (UCTP) is a well-known NP-hard combinatorial optimization problem involving the assignment of courses to timeslots, rooms, and professors under various institutional and resource constraints

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

[1]. Its complexity, coupled with evolving academic requirements, has driven extensive research across exact, heuristic, and hybrid approaches [2][3]. Traditional manual scheduling methods are increasingly inadequate for handling modern timetabling demands, including fairness, flexibility, and scalability.

Constraint Programming (CP) and mixed-integer programming provide expressive frameworks for modeling complex UCTP variants [4][5]. Meanwhile, advancements in educational AI have introduced intelligent recommendation systems, explainable planning tools, and conversational agents for academic support [7][8][9]. However, these solutions are typically fragmented and lack integration into a unified system.

While constraint programming and mathematical optimization provide expressive modeling capabilities, and recent advances in educational AI enable intelligent recommendation and conversational support, existing solutions remain largely fragmented. In practice, institutions require integrated systems that combine optimization, data acquisition, and decision-support functionalities within a unified workflow.

To address this gap, we present MounouZinc, an extensible timetabling and academic decision-support system that tightly integrates constraint-based optimization, AI-driven recommendation, and conversational guidance within a unified architecture. Built on MiniZinc, the system supports flexible constraint modeling, modular solver configurations, and seamless integration with a PostgreSQL backend. It features a web-based interface for capturing professor availability, an AI-driven professor–course matching module leveraging semantic similarity between faculty expertise and course descriptions, and a conversational chatbot for academic advising and FAQ support.

MounouZinc provides a unified framework that bridges optimization techniques with institutional workflows and AI-driven services, serving both as a research platform and a deployable solution for real-world academic environments. The proposed framework adopts a modular MiniZinc-based modeling approach that enables dynamic activation of constraints and solver interchangeability, facilitating adaptability to diverse institutional requirements. Furthermore, MounouZinc implements an end-to-end automated workflow encompassing data acquisition, preference modeling, optimization, validation, and deployment. Experimental evaluation demonstrates the system's feasibility and scalability, along with measurable improvements in solution quality compared to baseline approaches.

2 Related Work

Research on the University Course Timetabling Problem (UCTP) has a long history, with surveys and case studies highlighting its combinatorial complexity and diverse institutional requirements. Early surveys such as [3] synthesize exact, heuristic, and metaheuristic approaches, while more recent work [2] provides updated taxonomies of models, objectives, and datasets, emphasizing trends such as robustness, fairness, and hybrid optimization methods. Educational timetabling

research spans exact, heuristic, and hybrid methods, with recent surveys consolidating formulations and benchmarks across curriculum-, course-, and resource-based variants. In particular, [10] unifies problem definitions and benchmark suites, reinforcing the importance of standard testbeds for reproducible evaluation.

2.1 Real-World UCTP formulation

Recent works focus on rich, real-world UCTP formulations aligned with practical requirements. For instance, [4] proposes a multi-objective mixed-integer programming model balancing institutional criteria, while [11] addresses accreditation-constrained timetabling using MIP and heuristics. Robust timetabling under disruptions is studied in [12], and [5] presents a detailed case study using modern constraint- and mathematical-programming techniques, emphasizing transparency and maintainability.

2.2 Flexible and automated timetabling systems

There is growing interest in flexible and automated timetabling systems. A system in [6] combines mathematical modeling with an implementation-oriented architecture to support deployment in institutional environments. Similarly, [13] addresses business-school timetabling with multi-section courses, room stability, and professor preferences, highlighting the need for unified models integrating institutional policies and constraints. These align with MounouZinc's focus on modular design and system integration.

2.3 AI-driven course recommendation and academic planning

Another research strand focuses on AI-driven course recommendation and academic planning. Studies show that recommender systems can support course selection using performance, interests, and program requirements [7]. Broader perspectives on AI-based recommendation are discussed in [14], while [15] explores machine- and deep-learning models for academic planning. Recent work [8] emphasizes explainable and LLM-based approaches. Unlike these student-focused systems, MounouZinc extends recommendation to professor–course matching based on specialization and course characteristics.

2.4 Chatbots and conversational agents in higher education

A third line of research concerns chatbots in higher education. Reviews such as [9] highlight their growing role in student support and administrative assistance. Empirical studies [16] report benefits in scalability and accessibility, alongside challenges in data quality. Work such as [17] shows chatbots are effective for routine advising tasks, and case studies [18] confirm their utility for FAQs and student guidance. Unlike standalone chatbot systems, MounouZinc integrates conversational support directly into timetabling and course management workflows.

2.5 Research gap

Existing literature provides strong but largely fragmented contributions across timetabling, AI-based recommendation, and academic chatbot systems; however, no unified platform simultaneously integrates MiniZinc-based timetabling, AI-driven professor–course matching, structured preference collection, and conversational support. MounouZinc addresses this gap by delivering an end-to-end framework that combines optimization, recommendation, and interactive guidance within a single architecture, unifying paradigms that prior works treat in isolation—whether optimization-focused [4][5], recommendation-driven [7][8], or deployment-oriented systems lacking embedded intelligence and conversational capabilities [6][11]—thereby enabling both automated scheduling and enhanced, data-driven decision-making.

3 System Overview

This section provides an overview of the system's design principles, main system components, and typical workflow, illustrating how the tool supports the entire life-cycle of university course timetabling shown in Figure 1 from initial setup, data acquisition and preference collection, modeling, optimization, validation, timetable dissemination, and the intelligent student-facing iAdvisor module.

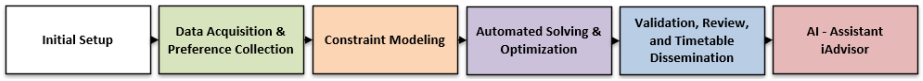


Fig. 1. Life Cycle

3.1 Design Principles

The development of MounouZinc was guided by four key goals that reflect the practical challenges faced by academic institutions:

1. **Modularity and Extensibility** Institutions differ widely in their organizational structures and scheduling policies. To accommodate this diversity, MounouZinc separates the data layer, constraint modeling layer, solver layer, and web interface. This modularity allows administrators to select the hard constraints to apply, incorporate the institutional rules by adjusting the weight of soft constraints, or switch solvers without modifying the core components.
2. **Usability for Non-Technical Staff** The scheduling is often carried out by administrative personnel rather than optimization experts. MounouZinc provides intuitive web interfaces for editing course data, capturing professor availability, visualizing schedules, and identifying constraints. The system is

designed so that most tasks require no direct interaction with MiniZinc or solver code. Everything can be handled through the web application.
3. **Data-Driven and Institution-Integrated Operation** Real deployments require synchronization with student information systems, HR data, and course catalogs. MounouZinc emphasizes structured data import and supports integration via APIs, CSV uploads, or database connectors.
4. **Decision Support** Beyond generating feasible schedules, MounouZinc offers an AI-based professor–course matching assistant. This assists planners in making informed adjustments and improves transparency for stakeholders. In addition, the AI chatbot will help students better select the courses in which to enroll.

3.2 Main System Components

As shown in Figure 2, MounouZinc is organized into five primary components that together support a complete end-to-end university timetabling workflow. These components define the functional structure of the system and establish the foundation upon which the architectural and implementation details in section 4 are built.

1. **Data and Persistence Layer (Postgres-Based)** This layer provides authoritative storage for all institutional data required during timetable generation. A normalized PostgreSQL schema maintains information on courses, professors, rooms, academic plans, institutional policies, and declared preferences. The schema ensures referential integrity, minimizes redundancy, and facilitates integration with Student Information Systems (SIS), HR databases, and curriculum management platforms. This layer serves as the central repository upon which all modeling and optimization tasks depend.
2. **Modeling and Solver Layer (MiniZinc-Based):** Institutional data is translated into MiniZinc models that encode both feasibility rules (hard constraints) and institutional preferences (soft constraints). The modeling layer is composed of modular constraint components covering professor availability, room compatibility, workload rules, accreditation requirements, and pedagogical structures. MiniZinc's solver-independent architecture enables MounouZinc to operate with multiple backends—including Chuffed, CP-SAT, and LCG-based solvers—allowing experimentation and solver substitution without changes to the underlying model.
3. **Web Interface and Interaction Layer (Node.js-Based):** The browser-based interface of the platform, powered by a Node.js and Express backend, enables administrators and professors to interact with the system intuitively and efficiently. Administrators can configure constraints, visualize timetables by room, professor, course, or study program, inspect violations, and manage scenario metadata. Professors use dedicated tools to declare availability, submit preferences, and review assigned teaching loads. Designed for non-technical users, this layer abstracts all modeling and solver complexity behind interactive workflows.

4. **AI-Assisted Services Layer:** MounouZinc incorporates intelligent components that extend its capabilities beyond traditional constraint solving by supporting decision-making, reducing administrative effort, and enhancing transparency for multiple stakeholders. These AI-driven features include preference ingestion mechanisms that transform availability and user inputs into structured model constraints, an NLP-based professor–course matching assistant that leverages teaching history, specialization data, and course descriptions to produce ranked assignment recommendations, and a Retrieval-Augmented Generation (RAG)-based conversational assistant that serves both as a FAQ and admissions tool for prospective students ("Chatbot") and as an academic advising agent ("iAdvisor") for current students, offering guidance on course selection, prerequisites, and enrollment constraints.
5. **Automation and Orchestration Layer (Python-Based):** A Python middleware coordinates data handling, model construction, solver invocation, and result processing. It generates MiniZinc instances from database records, assembles relevant constraint modules, executes solvers with configured strategies, parses solution outputs, and injects final schedules back into the database. This layer exposes APIs to the web interface and maintains detailed logs to ensure reproducibility, traceability, and operational reliability across scheduling cycles.

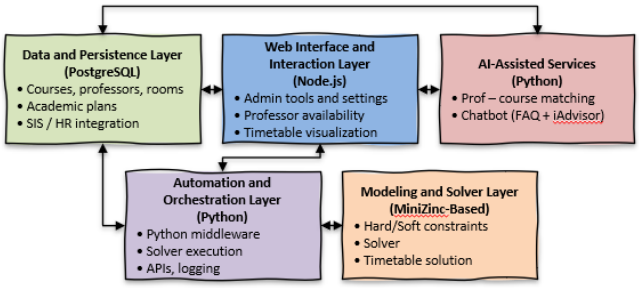


Fig. 2. High Level System Architecture

3.3 Typical Workflow

MounouZinc supports a structured and repeatable scheduling workflow aligned with real institutional practices. A typical cycle proceeds as follows:

1. **Data Import and Synchronization:** Administrators import course lists, rooms, professor profiles, timeslots, and academic plans from institutional databases or spreadsheets.

2. **Preference and Availability Collection:** Professors specify preferred teaching days and times. With the help of an AI-based tool, academic coordinators specify course-professor, and timeslot/room preferences.
3. **Constraint Configuration:** Administrators adjust institutional policies, enable optional constraint modules, assign penalty weights to soft constraints, and verify academic plan requirements or accreditation conditions.
4. **Model Generation and Solving:** The orchestration layer compiles all data and constraints into MiniZinc models. Solvers are executed under specified time limits or optimization goals. Final results are stored in the database.
5. **Visualization and Diagnostics:** Generated timetables along with statistical charts are visualized through the Web application. Users can access their own profile and check their schedule.
6. **Export and Deployment:** Final schedules can be exported as CSV, PDF, or Excel files, or published to institutional systems.
7. **Advising:** The chatbot and advising tools automatically update recommendations based on the finalized timetable and student's academic plan.

This workflow ensures that MounouZinc integrates seamlessly into academic planning procedures and supports continuous refinement throughout the scheduling period.

4 Architecture and Implementation

While Section 3 introduces the conceptual components of MounouZinc, this section describes their technical realization and the interactions between layers. MounouZinc is implemented using four core technologies—PostgreSQL, MiniZinc, Python, and Node.js—each mapped directly to the architectural layers of the system (see Figure 3). The architecture is fully modular, allowing institutions to configure, extend, or replace components as scheduling policies evolve.

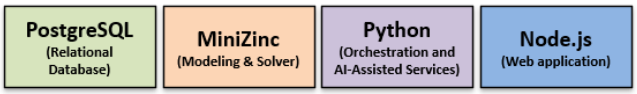


Fig. 3. Core Technologies

Figure 2 presents the high-level system architecture, organized into five layers: (4.1) data and persistence, (4.2) modeling and solver, (4.3) automation and orchestration, (4.4) web interface and interaction, and (4.5) AI-assisted services. Layers communicate through structured data schemas and APIs, ensuring separation of concerns and maintainability.

4.1 Data and Persistence Layer

The PostgreSQL database forms the foundation of the platform. Its relational schema includes tables for courses, professors, rooms, timeslots, departments, availability, workload, and academic plans. Foreign-key constraints enforce institutional consistency across departments.

4.2 Modeling and Solver Layer

Institutional data is mapped into MiniZinc decision variables, constraint modules, and optimization objectives. This modular design enables institutions and departments to activate only the constraint sets relevant to their specific rules, supporting flexibility and customization without altering the core engine. The modeling layer consists of:

1. *base domains* defining timeslots, professors, rooms, and timeslots;
2. *hard constraint modules* ensuring feasibility, including conflict prevention, room and professor availability;
3. *soft constraint modules* encoding preferences with adjustable weights for policies such as professor preferred teaching times.

4.3 Solver and Optimization Engine

MounouZinc uses MiniZinc as a solver-independent backend, supporting a variety of CP and hybrid solvers. Administrators can select the solver, adjust its time limits, and multi-objective weights reflecting institutional policy. Final solution is written back to the database and made available for review, visualization, and post-processing through the web interface. At this stage, the administrator can still perform changes on the generated timetable.

4.4 Automation and Orchestration Layer

The Python middleware automates the full data-to-solver pipeline. This layer ensures robust operation, reproducibility, and traceability across scheduling cycles, and enables integration with institutional workflows and external services. Responsibilities include:

1. generating MiniZinc data files from database content,
2. selecting and assembling appropriate constraint modules,
3. executing solvers and capturing logs,
4. parsing solution outputs, and
5. providing API endpoints for the web interface and external systems.

4.5 User Interface and AI Components

The professor–course matching module relies on transformer-based sentence embeddings to encode both course descriptions and faculty profiles into a shared semantic space. Similarity scores are computed using cosine similarity and used to rank candidate assignments. These scores are incorporated into the optimization model as weighted soft constraints, thereby influencing assignment decisions while preserving feasibility.

In parallel, the iAdvisor conversational agent leverages a Retrieval-Augmented Generation (RAG) architecture combining institutional regulations, academic catalogs, and student records. The system integrates rule-based validation with LLM-generated responses to ensure consistency with academic policies while providing flexible natural-language interaction.

5 Features and Usage

MounouZinc provides a comprehensive suite of features that support administrators, professors, and students throughout the scheduling lifecycle. Its modular design enables institutions to tailor the system according to their operational requirements while preserving a unified workflow.

5.1 Constraint Definition and Customization

Administrators can define and modify both hard and soft constraints through structured interfaces. Hard constraints (HC) must be strictly satisfied to ensure feasibility, while soft constraints (SC) are associated with weighted penalties, allowing fine-grained control over the overall quality of the generated schedule. All constraints are automatically translated into MiniZinc modules during model generation. Constraint activation is controlled through a dedicated database table, which specifies which modules should be included in a given MiniZinc model. This table is populated dynamically based on user input provided through the web application. The following list presents the different constraints considered:

1. HC: No two sessions can be scheduled in the same room at the same timeslot.
2. HC: Assigned professor must be qualified to teach the course.
3. HC: Assigned professor must be available during the session's timeslot.
4. HC: Professor must be assigned only one session in the same timeslot.
5. HC: Professors' total credit load must not exceed their maximum load.
6. SC: Prefer assigning courses to higher-priority timeslots.
7. SC: Prefer assigning courses to higher-priority rooms.
8. SC: Prefer assigning professors with higher suitability (course preference).
9. SC: Prefer assigning professors to their preferred timeslots.
10. SC: (Optional) Minimize total number of rooms used.

5.2 User Workflows

1. **Administrators** manage the timetabling system by importing data, maintaining core entities (professors, rooms, courses, timeslots), defining preferences and constraints, configuring the MiniZinc solver, running optimization, reviewing and updating the generated timetable if needed.
2. **Coordinators** manage departmental data by maintaining professors and courses, defining professor-course assignments, and verifying professors' time preferences.
3. **Professors** use a dedicated portal to specify their availability and preferred teaching slots. They can also view their assigned timetable, including courses, rooms, and teaching load, along with relevant statistics.
4. **Students** interact with the system through the iAdvisor chatbot to obtain information about degrees and majors and receive course recommendations based on prerequisites and their academic plan.

5.3 Visualization, Diagnostics, and Integration

MounouZinc provides an interactive timetable viewer that supports multiple perspectives, including overall schedule views, professor- and department-based views, and section-level views, complemented by charts presenting various statistical insights. All generated views can be exported in CSV format for further analysis and reporting. In addition, the system supports deployment either on-premise or within private cloud environments and integrates seamlessly with institutional SIS APIs, HR systems for managing professor data, and LMS platforms for schedule publication.This combined capability enables institutions to monitor, analyze, and distribute schedules efficiently while embedding MounouZinc into existing academic workflows with minimal operational overhead. These embedded features translate into stakeholder gains: administrators benefit from clearer constraint management, faculty members from a preference-entry portal and AI-assisted for availability, teaching wishes, expertise and students from an intelligent advising interface that guides their enrollment decisions and course selection.

6 Conclusion

This paper introduced MounouZinc, a comprehensive MiniZinc-based framework for university course timetabling that integrates optimization, data collection, recommendation, and conversational support into a single cohesive system. Unlike existing UCTP solutions—which typically focus solely on optimization or scheduling—MounouZinc extends functionality through its web interface for professor availability, AI-based professor–course matching, and a university chatbot capable of supporting prospective and current students. Experimental evaluation was conducted on synthetically generated datasets reflecting realistic institutional scenarios, ranging from small instances (3 professors, 5 sessions per

week) to large-scale ones (44 professors, 243 sessions per week), MounouZinc consistently produced feasible, high-quality timetables, demonstrating both the scalability of the framework and the inherent hardness of large UCTP instances. Beyond raw performance, the evaluation highlighted concrete benefits for stakeholder groups. Institutions benefit from reproducible workflows, automated data pipelines, and integrated student-support services. Future work will focus on hybrid constraint programming and machine learning approaches and expanded constraint modules for cross-campus coordination, advanced LLM-based advising, and multi-objective metrics for workload distribution. MounouZinc aims to serve as both a practical institutional tool and a research platform that advances the state of the art in academic scheduling and educational decision-support systems.

References

1. A. Schaerf, "A Survey of Automated Timetabling," The Artificial Intelligence Review, vol. 13, no. 2, pp. 87–127, Apr. 1999, doi: 10.1023/A:1006576209967.
2. M. C. Chen, S. N. Sze, S. L. Goh, N. R. Sabar, and G. Kendall, "A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities," IEEE Access, vol. 9, pp. 106515–106529, 2021, doi: 10.1109/ACCESS.2021.3100613.
3. H. Babaei, J. Karimpour, and A. Hadidi, "A survey of approaches for university course timetabling problem," Computers & Industrial Engineering, vol. 86, pp. 43–59, Aug. 2015, doi: 10.1016/j.cie.2014.11.010.
4. M. Mokhtari, M. Vaziri Sarashk, M. Asadpour, N. Saeidi, and O. Boyer, "Developing a Model for the University Course Timetabling Problem: A Case Study," Complexity, vol. 2021, pp. 1–12, Dec. 2021, doi: 10.1155/2021/9940866.
5. M. Davison, A. Kheiri, and K. G. Zografos, "Modelling and solving the university course timetabling problem with hybrid teaching considerations," J Sched, vol. 28, no. 2, pp. 195–215, Apr. 2025, doi: 10.1007/s10951-024-00817-w.
6. A. Muklason et al., "Flexible Automated Course Timetabling System with Lecturer Preferences Using Hyper-heuristic Algorithm: 7th International Conference on Sustainable Information Engineering and Technology, SIET 2022," SIET 2022 - Proceedings of 7th International Conference on Sustainable Information Engineering and Technology 2022, pp. 258–262, Nov. 2022, doi: 10.1145/3568231.3568273.
7. S. Cha, M. Loeser, and K. Seo, "The Impact of AI-Based Course-Recommender System on Students' Course-Selection Decision-Making Process," Applied Sciences, vol. 14, no. 9, p. 3672, Jan. 2024, doi: 10.3390/app14093672.
8. Jiawei Li, Qianru Lyu, Wei Qiu, and Andy W. H. Khong, "Improving Course Recommendation Systems with Explainable AI: LLM-Based Frameworks and Evaluations," July 2025, doi: 10.5281/ZENODO.15870185.
9. K. Peyton, S. Unnikrishnan, and B. Mulligan, "A review of university chatbots for student support: FAQs and beyond," Discov Educ, vol. 4, no. 1, p. 21, Jan. 2025, doi: 10.1007/s44217-025-00397-7.
10. S. Ceschia, L. D. Gaspero, and A. Schaerf, "Educational Timetabling: Problems, Benchmarks, and State-of-the-Art Results," European Journal of Operational Research, vol. 308, no. 1, pp. 1–18, July 2023, doi: 10.1016/j.ejor.2022.07.011.
11. N. M. Arratia-Martinez, P. A. Avila-Torres, and J. C. Trujillo-Reyes, "Solving a University Course Timetabling Problem Based on AACSB Policies," Mathematics, vol. 9, no. 19, Art. no. 19, Jan. 2021, doi: 10.3390/math9192500.

12. A. Gülcü and C. Akkan, "Robust university course timetabling problem subject to single and multiple disruptions," European Journal of Operational Research, vol. 283, no. 2, pp. 630–646, June 2020, doi: 10.1016/j.ejor.2019.11.024.
13. A. Ozkan, A. Ulucan, C. Dirik, and K. B. Atici, "University course timetabling with multi-section courses, room stability and professor preferences: an application in a business school," Comput Manag Sci, vol. 22, no. 1, p. 3, Feb. 2025, doi: 10.1007/s10287-025-00529-2.
14. Z. Akbar, E. Sopandi, B. Badruzzaman, and M. Khalik, "The Role of Artificial Intelligence-Based Recommendation Systems in Selection of Courses for Students," Journal of Social Science Utilizing Technology, vol. 1, pp. 249–260, Dec. 2023, doi: 10.70177/jssut.v1i4.671.
15. A. Kord, A. Aboelfetouh, and S. M. Shohieb, "Academic course planning recommendation and students' performance prediction multi-modal based on educational data mining techniques," J Comput High Educ, Jan. 2025, doi: 10.1007/s12528-024-09426-0.
16. M. Dawood, "Assessing the effectiveness of Chatbots in providing personalized academic advising and support to higher education students: A narrative literature review," Studies in Technology Enhanced Learning, vol. 4, no. 1, Oct. 2024, doi: 10.21428/8c225f6e.7140f8f4.
17. M. M. Thottoli, B. H. Alruqaishi, and A. Soosaimanickam, "Robo academic advisor: Can chatbots and artificial intelligence replace human interaction?," CONT ED TECHNOLOGY, vol. 16, no. 1, p. ep485, Jan. 2024, doi: 10.30935/cedtech/13948.
18. B. R. Ranoliya, N. Raghuwanshi, and S. Singh, "Chatbot for university related FAQs," in 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi: IEEE, Sept. 2017, pp. 1525–1530. doi: 10.1109/ICACCI.2017.8126057.

Fast roster generation for single-nurse rostering problems

Robin Tourlamain[1,2][0009−0003−3782−8063], Pieter Smet[2][0000−0002−3955−7725], and Greet Vanden Berghe[2][0000−0002−0275−5568]

[1] The University of Melbourne, School of Mathematics and Statistics
Parkville, VIC 3010, Australia
[2] KU Leuven, Department of Computer Science, NUMA
Gebroeders de Smetstraat 1, 9000 Gent, Belgium
{robin.tourlamain, greet.vanden.berghe, pieter.smet}@kuleuven.be

Abstract. Despite significant research into nurse rostering problems, generating feasible solutions for large instances that include complex hard constraints as part of nurses' contracts, and proving their optimality, remains a challenge. The combination of long rostering periods and many shift types leads to enormous search trees that simply cannot be fathomed by current exact algorithms. In this paper we help close this gap by developing effective pruning strategies for dynamic programming approaches, drastically reducing the number of individual rosters that must be considered, as well as a heuristic that quickly identifies feasible rosters. We demonstrate how incorporating these strategies into state-of-the-art methods significantly speeds up the computation of optimal (partial) solutions. The strategies we suggest for rostering a single nurse in isolation can be embedded within a complete nurse rostering approach.

Keywords: Nurse rostering · Dynamic programming · Pruning strategies

1 Introduction

Nurse Rostering Problems (NRPs) require the assignment of shifts to nurses throughout a given rostering period so that shift staffing demand is met while respecting various constraints that arise from the nurses' contracts. Numerous variants of these problems have been studied, differing in terms of their objectives and the extent to which certain constraints are relaxed. This paper focuses on a specific NRP first introduced by Curtois et al. [6], which distinguishes itself from other NRPs by featuring large rostering periods of up to one year and many hard constraints at the single-nurse level. This complicates the generation of optimal or simply just feasible rosters for each individual nurse, a problem we will refer to as the single-nurse rostering problem (SNRP).

Exact methods for this specific NRP rely on decomposing the problem into multiple SNRPs [4,12], where each SNRP is solved using Dynamic Programming (DP). However, even the best algorithms currently available in the academic literature are unable to obtain optimal solutions for large instances of this problem.

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

We propose an extension of the DP algorithm for the SNRP described by Legrain et al. [12] that enables stronger pruning of the roster solution space, thereby accelerating the search for optimal rosters. Additionally, we introduce a guiding heuristic that efficiently generates feasible rosters for the largest SNRPs while preserving the guarantee of global optimality.

The remainder of this paper is structured as follows. First, we review the related literature on SNRPs in Section 2. Next, Section 3 presents a definition of the specific NRP we will address along with a brief review of the model for which we will develop our algorithmic improvements. In Section 4 we introduce various strategies for accelerating optimal roster generation, while Section 5 focuses on finding feasible solutions quickly. A computational study in which we quantify the impact of our contributions is presented in Section 6. Finally, Section 7 concludes the paper.

2 Related work

Although the SNRP has received limited attention as a standalone problem, it is often studied when decomposing a NRP. For example, Aickelin et al. [1] first use a genetic algorithm to construct a set of feasible rosters for each nurse, before assembling a combination into the full roster. Although many studies adopt this two-phase approach, the methods used to solve the SNRPs differ considerably. For example, Demassey et al. [7] generate single-nurse rosters using constraint programming, while Zhang et al. [18] formulate the SNRP as an integer programming network flow model. The winners of the second International Nurse Rostering Competition [5] also developed a network flow formulation in which the nurses' contractual constraints were modeled as part of the network structure [15]. The SNRP can itself be further decomposed, as demonstrated by Brucker et al. [3], who generate short feasible shift sequences and then combine these into a single-nurse roster using greedy heuristics. Legrain et al. [13] apply a similar approach and construct the final roster using a network flow integer programming formulation.

One approach to generate single-nurse rosters that is effective in practice is to reformulate the SNRP as a Shortest Path Problem with Resource Constraints (SPPRC), where each complete path corresponds to a feasible roster for a single nurse. Jaumard et al. [11] first generate all feasible paths and apply a standard shortest path algorithm to find an optimal solution. Maenhout et al. [14] improve upon this method by introducing dominance rules, eliminating the need to explicitly enumerate all feasible paths. Burke et al. [4] further reduce computation time through a beam search strategy, resorting to a full tree search only if no desirable roster is found. Strandmark et al. [16] show that solving the SPPRC to optimality is not necessary to produce high-quality rosters using their heuristic method. Dohn et al. [9] decompose the SNRP and solve the SPPRC over a set of work stretches instead of individual shifts.

The state-of-the-art approach for NRPs was developed by Legrain et al. [12], who have the best results for multiple nurse rostering datasets using their

SPPRC-based branch-and-price method. They were able to achieve this by introducing *strong* dominance rules in their DP algorithm for the SNRP, which not only consider the current state of a partial roster, but also the remaining possible completions. However, while they employ various acceleration techniques such as bidirectional search and beam search, solving the largest SNRPs remains a computational challenge.

3 Problem definition

This section begins by describing the considered NRP in Section 3.1, before moving on to the SPPRC model of the SNRP which forms the basis of the current best algorithm [12] in Section 3.2.

3.1 Nurse rostering

Curtois et al. [6] introduce a NRP where nurses are limited to at most one shift per day. The roster is subject to a variety of constraints which are categorized as either hard (violations forbidden) or soft (violations penalized), with the objective being to minimize the total sum of penalties.

For each day and shift type, soft coverage constraints indicate the minimum number of worked shifts required across all nurses as well as a maximum number to avoid over-staffing.

Furthermore, each nurse has a personalized contract that enforces a set of individual constraints which are especially relevant for the SNRP. Bilgin et al. [2] identify three types of nurse-specific constraints and group them based on the purpose they serve in the roster. Two of these constraint types appear in the NRP from Curtois et al. [6]: *counter* and *series* constraints. Each nurse's contract contains hard counter constraints, limiting the total number of hours or weekends a nurse may work. A contract may also specify individual counter constraints for each shift type. Additionally, each nurse is subject to series constraints which dictate a minimum and maximum number of consecutive days they may work. A second series constraint imposes a minimum number of consecutive days idle, with no upper limit. Each nurse is also assigned a unique set of days on which they cannot work any shift and must remain idle.

All nurses must respect hard shift succession constraints that forbid them to work certain shifts immediately after another. Finally, a nurse may indicate preferences to work or not work certain shifts on certain days. Satisfying these preferences is considered part of the objective.

3.2 SPPRC model

The improvements we will propose in Section 4 are based on a common SPPRC formulation of the SNRP [4,16,12] that Legrain et al. [12] described clearly. Therefore, in this section we will restrict ourselves to providing a brief overview of relevant notation and the most important concepts of the SPPRC model.

The single-roster search space is represented by a directed acyclic graph $G = (V, A)$, with a set of vertices V and set of arcs A. Each vertex v represents a unique day-shift combination, with arcs corresponding to the valid transitions between them. Arcs flow from an artificial source o to an artificial sink t. Graph G in Figure 1 provides an example that features a period of four days (d) and two shift types (s), where vertices at equal distance from o represent the distinct shift types available on that day. For each day, a dummy vertex s_0 is introduced to represent the absence of a worked shift. Constraints are modeled as bounded resources that are consumed when traversing certain arcs, with R denoting the set of all resources. Let P be a partial path from o to v. For any partial path P in G, the consumption of such a resource r is denoted $\gamma^r(P)$. For example, to enforce a constraint on the maximum number of hours worked, a bounded resource r_w is defined that must satisfy $\gamma^{r_w}(P) \leq U^{r_w}$, where U^{r_w} represents the maximum number of hours worked. Upon entering a vertex v, the number of hours in the corresponding shift type is added to $\gamma^{r_w}(P)$. A complete and feasible roster is any path from o to t that satisfies all resource constraints. The cost of a path P, denoted $\gamma^0(P)$, is defined by the sum of all penalties incurred by violations of soft constraints. A completion of P is a path from o to t that includes P itself and is denoted as $[P, \overline{P}]$, where $\overline{P}$ is the path from v to t. We refer to $\overline{P}$ as the suffix.

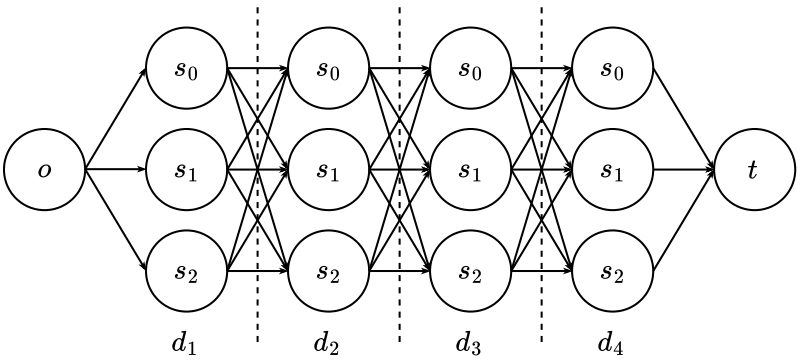


Fig. 1. Example of the graph G for a SNRP instance with a rostering period of four days and two shift types.

Building a solution involves iteratively extending partial paths along the arcs, starting from source o. With every extension, the resources consumed along the path are updated using Resource Extension Functions (REFs), and the cost is adjusted based on Cost Extension Functions (CEFs). Desaulniers et al. [8] employed this terminology in the context of vehicle routing, with Legrain et al. [12] later formalizing it for the NRP. We refer to Legrain et al. [12] for detailed definitions of the REFs and CEFs of the SNRP we are addressing in this paper. The feasible path from o to t with the minimum cost corresponds to an optimal single-nurse roster.

4 Partial path dominance

In practice, extending all feasible paths through graph G is intractable for large instances due to computation time or hardware limitations. Fortunately, we can prove that some partial paths are dominated and can thus be safely pruned without ever eliminating the optimal solution. The general dominance criterion is defined by Legrain et al. [12] as follows:

Dominance: *Let P and Q be two distinct paths from origin o to a vertex v. Path P dominates Q if and only if any feasible completion of Q, $[Q, \overline{Q}]$, yields a feasible completion of P, $[P, \overline{Q}]$, such that $\gamma^0\left([P, \overline{Q}]\right) \leq \gamma^0\left([Q, \overline{Q}]\right)$, and if there is a feasible completion of Q, $[Q, \tilde{Q}]$ such that $\gamma^0\left([P, \tilde{Q}]\right) < \gamma^0\left([Q, \tilde{Q}]\right)$.*

For a path P to dominate another path Q, their resources must satisfy a set of *dominance rules*. Let L^r and U^r denote the lower and upper bounds on resource $r \in R$. For example, for a series constraint on consecutive days worked, L^r and U^r correspond to the minimum and maximum allowed number of feasible consecutive assignments. Let $\overline{D}$ be the number of days remaining in the rostering period after vertex v, and let us define $[x]^+ = \max(x, 0)$. The dominance rules state that P dominates Q with respect to resource r if:

$$\left[\gamma^r(P) + \overline{D} - U^r\right]^+ \leq \left[\gamma^r(Q) + \overline{D} - U^r\right]^+ \text{ and } [L^r - \gamma^r(P)]^+ \leq [L^r - \gamma^r(Q)]^+ \quad (1)$$

Legrain et al. [12] conducted extensive computational experiments with various acceleration strategies for DP and found that bidirectional path extension is the most effective for reducing computation time. In the following sections, we will extend their bidirectional dominance-based path extension algorithm to improve its performance.

4.1 Inter-shift dominance (ISD)

Two partial rosters can only be compared using dominance rules (1) if they both terminate at the same vertex, meaning they must end with the same shift type. This rule originates from vehicle routing problems [10], where vertices represent locations. However for the SNRP, this needlessly excludes other valid comparisons. We will therefore introduce a more precise rule to facilitate *inter-shift dominance*.

Recall that for P to dominate Q, all feasible completions of Q must also be feasible completions of P. The identical vertex criterion simply avoids that $\overline{Q}$ is an infeasible completion of P due to a forbidden shift succession between the last shift of P and the first shift of $\overline{Q}$. However, we can guarantee this property at the shift type level. Indeed, for this property to hold it suffices that the set of feasible shift successions for P includes at least those that are feasible for Q.

Let $j \in V$ be a vertex corresponding to a certain shift type. The set of neighbouring vertices corresponds to the set of feasible shift successions and is defined as $N^+(j) = \{k \in V \mid (j, k) \in A\}$. Let P be a partial path from origin o

to u, while Q is a partial path from o to v. P dominates Q with respect to the succession constraints if:

$$N^+(v) \subseteq N^+(u)$$

4.2 Maximum remaining resource consumption (MRC)

The efficacy of dominance rules (1) can be improved by substituting $\overline{D}$ with a better approximation of the resource's maximum remaining consumption (MRC). As $\overline{D}$ gets smaller, more paths become equivalent with respect to r, allowing a larger number of paths to be pruned away. In what follows, we will define improved approximations for several constraint-associated resources by incorporating information from the relevant constraint parameters. Note that the approximated MRC at a vertex v must not underestimate the true MRC, which ensures that paths which could lead to a better complete roster will not be pruned.

Maximum number of consecutive days worked The MRC of this constraint's resource is equal to $\overline{D}$ under the assumption that the nurse in question is available for work during the remainder of the period. However, many rosters include fixed days off, forcing the stretch of days worked to end prematurely. Thus, we replace $\overline{D}$ with the number of days remaining until the next fixed day off.

Workload constraints For constraints concerning the number of days worked and the number of assignments of a specific shift type, the MRC is closely related to the maximum number of days available for work in the remainder of the rostering period. We can therefore approximate the MRC of this resource.

The presence of hard constraints on the maximum number of consecutive days worked implies that, in most cases, the actual maximum number of days worked in any suffix is less than $\overline{D}$. Consider a pattern with length $B = U^{\text{work}} + L^{\text{off}}$ defined by working the maximum number of consecutive days U^{work} followed by the minimum number of consecutive days off L^{off}. Repeating this pattern for the remaining rostering period provides a good approximation of resource r's MRC.

For an arbitrary number of remaining days $\overline{D}$, we first compute the number of full patterns f that fit within this period, the number of trailing days t after working those patterns, and the subset of those days t_w available for work. The approximation of the MRC of working days MRC^r is:

$$f = \left\lfloor \frac{\overline{D}}{B} \right\rfloor, \quad t = \overline{D} \bmod B, \quad t_w = \min(t, U^{\text{work}})$$

$$MRC^r = f \cdot U^{\text{work}} + t_w \tag{2}$$

In some cases this approximation can be improved even further. Recall we are using a bidirectional extension algorithm, alternating between extending paths originating from the source (forward) and the sink (backward). During the backward pass, we are essentially computing the MRC of resources from

the forward pass, and vice versa. Let us therefore iteratively extend the forward and backward paths, with D corresponding to the total number of days in the rostering period. After I^f iterations of forward extension and I^b iterations of backwards extension, we can derive the MRC of the forward resources for the final I^b days. Simply iterate over the resource vectors of partial paths stored at day $D - I^b$ and extract the maximum value for each resource.

However, the forward and backward paths are still separated by $D - I^f - I^b$ days. We therefore combine the two approximations to find the most accurate MRC per resource. This process begins by taking the maximum consumption of a resource in paths emerging from the opposite direction. We then approximate the maximum number of remaining workdays in the $D - I^f - I^b$ days separating the opposite passes using Equation (2). The sum of these values is the resource's final MRC. To reduce computational overhead, we do not compute these values for each label we extend, but only once for all labels corresponding to the same day in the roster.

4.3 Pre-dominance pruning (PDP)

Another well-known technique to accelerate dominance is to remove as many partial paths as possible in advance. Curtois et al. [4] remove equivalent paths, while Legrain et al. [12] extended this by also pruning labels that can never reach the desired objective value. They achieve this by executing a shortest reverse path search from the sink to obtain lower bounds on completions for each vertex.

Similarly, an upper bound on a resource's remaining consumption can be used to prune paths with a constraint-imposed minimum consumption. Instead of utilizing a dedicated longest path algorithm to generate upper bounds, we simply reuse the MRC from Section 4.2 to prune away paths before the dominance procedure. More formally, a path P can be discarded if for any $r \in R$ the inequality $\gamma^r(P) + MRC^r < L^r$ holds.

4.4 Dynamic bidirectional extension (DBIDI)

The bidirectional extension algorithm extends partial paths starting at both source o and sink t. This is an effective acceleration strategy as it decreases the depth of the search tree and thus significantly shrinks its overall size. One parameter to consider is the point where both passes eventually meet.

Legrain et al. [12] chose the middle of the rostering period as a natural meeting point. However, research into bidirectional extension algorithms for general SPPRCs suggests that asymmetry between the forward and backward search trees can significantly impact performance when a static midpoint is used [17]. In the SNRP, fixed days off can contribute to this asymmetry, as the number of propagated labels remains constant on those days. This in turn result in a higher number of propagated paths in either the forward or backward pass.

We therefore extend the forward or backward pass depending on which currently has fewer paths to extend, following a strategy proposed by Tilk et al. [17]. All paths are then merged once they inevitably meet.

5 Fast feasibility

Constructing an optimal single-nurse roster requires identifying the lowest-cost complete path by extending and dominating partial paths until the end of the rostering period, thereby implicitly enumerating all feasible rosters. Even with the improvements introduced in Section 4, this process can remain computationally intensive, particularly for large problem instances. Indeed, just finding a feasible solution for these instances can take considerable time. In practice, however, feasible rosters must be generated quickly.

We have identified two primary factors contributing to long computation times for large SNRP instances: (i) hard workload constraints, which make only near-complete paths feasible, and (ii) a high branching factor in the search trees due to the large number of shift types. Additional complications arise from hard constraints on consecutive days worked or off, combined with long rostering periods.

Burke et al. [4] and Legrain et al. [12] accelerate the generation of feasible rosters by limiting the number of propagated paths. The choice of which paths to propagate is crucial for the effectiveness of this beam search method. For example, Legrain et al. [12] only extend the ten paths with the lowest objective values in each iteration. We propose an alternative path selection heuristic designed for quickly producing feasible solutions. To this end, we introduce the concept of *expected workload*: for each day in the rostering period, this quantity approximates the cumulative number of hours a nurse should have worked by the end of that day in order to satisfy the workload constraints by the end of the period.

Let D denote the number of days in the rostering period and W the number of weekends. A subset of those days and weekends are not available for work due to the fixed idle days defined in the constraints. We refer to those that are available as D_a and W_a. Additionally, we denote the subset of days that are part of a weekend as D_w. Furthermore, some non-fixed weekend days will not be worked due to counter constraints on weekends. Let U^{weekend} be the maximum number of weekends a nurse can work. The average number of weekend days available in a feasible roster is then:

$$\langle D_{a,w} \rangle = \frac{D_{a,w}}{W_a} \cdot U^{\text{weekend}}$$

From this, the average number of workdays available in the rostering period is:

$$\langle D_a \rangle = D_a - D_{a,w} + \langle D_{a,w} \rangle$$

Let U^{workload} be the maximum number of minutes a nurse can work. The average load per available work day is defined by ℓ. Multiplying ℓ with the average number of available weekend days gives the total average workload on weekends ℓ_w:

$$\ell = \frac{U^{\text{workload}}}{\langle D_a \rangle}, \quad \ell_w = \ell \cdot \langle D_{a,w} \rangle$$

Since the specific weekends worked are not known in advance, this load is distributed evenly across all non-fixed weekend days. The per-day workload estimates are therefore:

$$\ell_{\text{weekday}} = \ell \text{ and } \ell_{\text{weekend}} = \frac{\ell_w}{D_{a,w}}$$

The expected workload for each day in the rostering period is computed as the cumulative sum of these daily estimates, ignoring fixed days off. During path selection, a number of paths whose cumulative workloads are close to the expected workloads are selected first to avoid paths becoming infeasible before the end of the rostering period. If this process does not result in a feasible path on a first pass from source o to sink t, we increase the number of propagated paths and perform a new extension pass.

6 Computational study

This section assesses the impact of our improved dominance rules and modified bidirectional search on the performance of the state-of-the-art algorithm introduced by Legrain et al. [12]. We enable each improvement from Section 4 individually and compare the results to those of our implementation of the original algorithm. We also evaluate the performance of the algorithm with all improvements enabled. Additionally, we measure the time it takes to produce an initial feasible solution for each instance using our expected workload path selection criterion from Section 5 while restricting the initial number of propagated paths to 64. We chose this number after experimentation and found it to be fast to compute for all different lengths of rostering periods.

For benchmarking, we used a dataset from the literature consisting of 24 instances [6]. Each instance contains as many SNRPs as there are nurses. To better reflect real-world usage of the algorithms, we solve all SNRPs in an instance 20 times, each time with a different vector of dual prices representing the value of covering a specific shift that is also part of the objective function. These dual prices were obtained by executing a column generation algorithm on each instance, and recording the dual prices in each iteration. The specific dual prices are available in a public repository[3]. The reported values correspond to the average time required to solve all SNRPs in an instance, averaged over these 20 repetitions.

Table 1 shows the reductions in computation time obtained by the improvements. The table only includes instances for which each algorithm could solve all SNRP subproblems. Negative values indicate a slowdown. The last column displays the time it took to generate a complete feasible roster with the expected workload criterion. The algorithms are implemented in the Rust programming language and experiments were run on a single core of an AMD Ryzen 9 5950X processor, with the maximum runtime set to five hours per subproblem.

[3] https://gitlab.kuleuven.be/codes/datasets/curtois2014-dual-prices

Strategy	Base	ISD	MRC	PDP	DBIDI	Combined	Feasible
Instance	Avg. time (s)	Average time reduction (%)					Time (s)
Instance 1	0.0005	-12.97	8.28	-32.97	-0.86	-9.35	0.0020
Instance 2	0.0011	0.45	23.56	8.36	13.41	26.20	0.0038
Instance 3	0.0059	14.14	23.43	9.15	17.08	29.50	0.0055
Instance 4	0.0145	6.00	49.35	31.57	5.63	53.51	0.0065
Instance 5	0.1649	-8.89	81.99	49.63	36.71	84.97	0.0106
Instance 6	0.0359	5.60	49.04	25.07	7.91	53.34	0.0164
Instance 7	0.1328	6.45	49.69	22.36	23.08	62.53	0.0159
Instance 8	16.1277	11.09	70.86	37.21	20.24	74.16	0.0228
Instance 9	0.5904	18.79	17.43	4.44	14.70	42.66	0.0333
Instance 10	3.9299	39.54	60.06	10.84	9.56	73.83	0.0367
Instance 11	0.2935	48.58	52.69	31.28	11.22	75.37	0.0565
Instance 12	1.0039	64.45	50.15	22.19	14.11	82.42	0.0997
Instance 14	1005.5752	50.50	95.91	28.40	53.96	97.95	0.1302
Instance 16	0.6222	23.94	65.29	73.15	18.70	81.78	0.0318
Instance 17	20.2740	13.93	64.90	75.72	21.78	87.57	0.0774
Instance 18	154.1834	2.39	97.03	75.58	42.51	98.34	0.0565
Instance 19	3658.5361	27.79	87.72	33.52	34.99	94.36	0.1129
Instance 20	840.3358	62.89	65.35	73.33	3.41	91.27	0.4132
Global avg.	-	**20.82**	**56.26**	**32.16**	**19.34**	**66.69**	-

Table 1. Comparison of experiments that yielded optimal solutions within 5 hours.

For each improved algorithm, the general trend is that they become more effective as the roster solution space grows. ISD peaks at a 64% improvement over the base algorithm for instance 12, which happens to be the instance with the most distinct shift types we were able to solve to optimality. Its overall improvement is similar to DBIDI at around 20%, but these methods are most effective for entirely different subproblems. Incorporating the MRC into the dominance rules provides the most effective improvement overall, with an average speedup of 56%. Combining all improvements yields an average speedup of 66% over all subproblems but exhibits a significantly larger speedup for the largest instances, with several reductions in computation time of over 90%.

Strategy	Base	ISD	MRC	PDP	DBIDI	Combined	Feasible
Instance	Number of subproblems solved						Time (s)
Instance 13	0	0	0	0	0	1	0.4040
Instance 15	10	18	20	10	20	20	0.0643
Instance 21	0	0	0	0	0	1	1.0277
Instance 22	0	0	0	0	0	0	1.4175
Instance 23	0	0	0	0	0	0	3.4389
Instance 24	0	0	0	0	0	0	33.1022

Table 2. Comparison of experiments that exceeded the 5 hours time limit.

In case one or more subproblems could not be solved by all algorithms, no relative improvement could be calculated. Instead, we compare the number of subproblems that could be solved to optimality in Table 2. These results show that our improvements can provide the additional efficiency that is needed to solve some of the larger subproblems. Indeed, either strengthening the dominance rules based on the projected MRC or merging paths at a dynamic midpoint enabled us to solve all subproblems associated with Instance 15. Combining all improvements enabled us to solve the first subproblem associated with Instance 13 and 21 to optimality. We should note that in these first subproblems, the dual prices associated with each shift are identical, enabling us to prune away many partial paths.

Although the method could not solve all subproblems to optimality, we did succeed in generating feasible solutions for all original NRP instances. This process took approximately 30 seconds for the largest instance, demonstrating the capabilities of the expected workload selection criterion with regards to generating feasible solutions.

7 Conclusion

In this paper we successfully accelerated the dominance-based path extension algorithm for SNRP introduced by Legrain et al. [12] by refining existing dominance rules and introducing new specialized pruning techniques. These improvements led to reductions in computation time of up to 98% and enabled us to solve instances larger than was possible before. Additionally, we introduced an efficient method to generate feasible rosters for the largest SNRP instances through heuristic path ordering, based on the *expected workload* of the partial paths.

Any algorithm for NRPs that relies on solving its constituent SNRPs as subproblems can benefit from our contributions, with the broader consequence of accelerating an entire family of NRP solution methods.

Acknowledgements This research was partially supported by KU Leuven, Belgium and The University of Melbourne, Australia through the KU Leuven - Melbourne Joint PhD program; by KU Leuven C24E/23/012 'Human-centred decision support based on new theory for personnel rostering' and Research Foundation — Flanders (FWO) under grant number K804824N.

References

1. Aickelin, U., Dowsland, K.A.: An indirect genetic algorithm for a nurse-scheduling problem. Computers & operations research **31**(5), 761–778 (2004)
2. Bilgin, B., De Causmaecker, P., Rossie, B., Vanden Berghe, G.: Local search neighbourhoods for dealing with a novel nurse rostering model. Annals of Operations Research **194**(1), 33–57 (2012)
3. Brucker, P., Burke, E.K., Curtois, T., Qu, R., Vanden Berghe, G.: A shift sequence based approach for nurse scheduling and a new benchmark dataset. Journal of Heuristics **16**(4), 559–573 (2010)

4. Burke, E.K., Curtois, T.: New approaches to nurse rostering benchmark instances. European Journal of Operational Research **237**(1), 71–81 (2014)
5. Ceschia, S., Dang, N., De Causmaecker, P., Haspeslagh, S., Schaerf, A.: The second international nurse rostering competition. Annals of Operations Research **274**(1), 171–186 (2019)
6. Curtois, T., Qu, R.: Computational results on new staff scheduling benchmark instances. ASAP Res. Group, School Comput. Sci., Univ. Nottingham, Nottingham, UK, Tech. Rep (2014)
7. Demassey, S., Pesant, G., Rousseau, L.M.: A cost-regular based hybrid column generation approach. Constraints **11**(4), 315–333 (2006)
8. Desaulniers, G., Desrosiers, J., loachim, I., Solomon, M.M., Soumis, F., Villeneuve, D.: A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In: Fleet management and logistics, pp. 57–93. Springer (1998)
9. Dohn, A., Mason, A.: Branch-and-price for staff rostering: An efficient implementation using generic programming and nested column generation. European Journal of Operational Research **230**(1), 157–169 (2013)
10. Gschwind, T., Irnich, S., Rothenbächer, A.K., Tilk, C.: Bidirectional labeling in column-generation algorithms for pickup-and-delivery problems. European Journal of Operational Research **266**(2), 521–530 (2018)
11. Jaumard, B., Semet, F., Vovor, T.: A generalized linear programming model for nurse scheduling. European journal of operational research **107**(1), 1–18 (1998)
12. Legrain, A., Omer, J.: A dedicated pricing algorithm to solve a large family of nurse scheduling problems with branch-and-price. INFORMS Journal on Computing **36**(4), 1108–1128 (2024)
13. Legrain, A., Omer, J., Rosat, S.: A rotation-based branch-and-price approach for the nurse scheduling problem. Mathematical Programming Computation **12**(3), 417–450 (2020)
14. Maenhout, B., Vanhoucke, M.: Branching strategies in a branch-and-price approach for a multiple objective nurse scheduling problem. Journal of scheduling **13**(1), 77–93 (2010)
15. Römer, M., Mellouli, T.: A direct milp approach based on state-expanded network flows and anticipation for multi-stage nurse rostering under uncertainty. In: Proceedings of the 11th international conference on the practice and theory of automated timetabling. pp. 549–551 (2016)
16. Strandmark, P., Qu, Y., Curtois, T.: First-order linear programming in a column generation-based heuristic approach to the nurse rostering problem. Computers & Operations Research **120**, 104945 (2020)
17. Tilk, C., Rothenbächer, A.K., Gschwind, T., Irnich, S.: Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. European Journal of Operational Research **261**(2), 530–539 (2017)
18. Zhang, H., Jackson, L., Tako, A., Liu, J., Dunnett, S.: Network based formulations for roster scheduling problems. In: Proceedings of the 11th International Conference on Practice and Theory of Automated Timetabling (PATAT 2016). pp. 403–419 (2016)

Assignment and Routing in the Home Health Care Problem*

Shihhung Su[1][0009−0005−4961−8746] and Chiuyuan Chen[2][0000−0002−3716−3810]

[1] National Yang Ming Chiao Tung University, Taiwan
shihong253@gmail.com
[2] National Yang Ming Chiao Tung University, Taiwan
cychen@nycu.edu.tw

Abstract. This study addresses the Home Health Care (HHC) problem with the goals of minimizing both the cost of hiring caregivers and the transportation cost incurred as caregivers travel between patients' geographic coordinates of residences, while satisfying patient-friendliness constraints. To achieve the aforementioned goals, this study proposes a new caregiver assignment and routing algorithm, called hiCH. Our algorithm is a heuristic algorithm. It does not require a set of caregivers to be given in advance, but instead hires caregivers one by one as the problem is addressed, thereby reducing hiring costs. Our algorithm hiCH adopts a hierarchical convex hull approach, considering the convex hull formed by unvisited patients and selecting a patient with the highest care level required on the convex hull to determine the employment level of the caregiver and serve as the starting point of the caregiver's visiting path, with the aim of reducing transportation cost. We improve hiCH by proposing D-hiCH, in which patients requests are further sorted according to their required service levels and time windows. To further improve solution quality, we also propose a genetic algorithm (GA) version of D-hiCH.

Keywords: Timetabling in Home Health Care (HHC) · assignment · routing · convex hull · genetic algorithm (GA).

1 Introduction

The aging population has become a pressing global concern, leading to a rapidly growing demand for medical services, particularly among individuals with chronic illnesses or limited mobility. In response to this challenge, Home Health Care (HHC) has emerged as a vital solution. HHC promotes care that is not only more accessible but often more cost-effective and better aligned with long-term health outcomes. See references [3] and [6].

The two primary components in an HHC system are: patients and caregivers. Patients submit requests for services that detail their specific care requirements

* This research was partially supported by the National Science and Technology Council of the Republic of China under the grant NSTC 113-2115-M-A49-012-MY2.

(or demands), such as the duration of the required service, the time window available for the assigned caregiver to visit, and the needed skill level of the assigned caregiver. The HHC problem focus on assigning all requests to sufficient number of caregivers and constructing feasible schedules for the caregivers that respect working time regulations, while at the same time, trying to minimize the cost caused by hiring caregivers and transportation between geographic coordinates of residences (or simply residences) of patients.

Based on the above, the two major decision-making aspects of the HHC problem are: assignment and routing. Assignment involves determining the number of caregivers required and allocating each patient's request to a suitable caregiver. Routing involves determining the sequence in which each caregiver visits the assigned patients. When modeling the above two major decision-making aspects, a distinction arises regarding how a caregiver's working time is defined.

Let $(*)$ denote the depot or the caregivers' residence. We say that a caregiver's working time is of *type 1* (respectively, *type 2*) if the travelling time from $(*)$ to the first patient and from the last patient back to $(*)$ is included (respectively, excluded) in the working time, as shown in Figure 1.

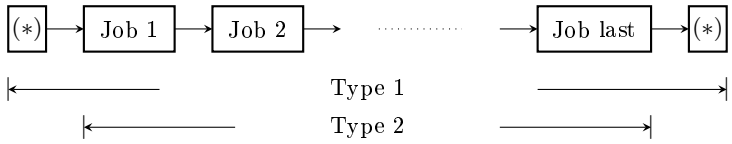


Fig. 1. Two types of working time: $(*)$ means the depot or the caregiver's residence.

The most related works of this manuscript are [1, 3, 5, 6]. We summarize these related works in Table 1 and briefly describe these works below.

Reference [3] considers weekly planning of HHC; and, the HHC problem is decomposed into four sequential sub-problems: (1) districting, (2) dimensioning, (3) assignment, and (4) routing. These four sub-problems are integrated and formulated as a mixed-integer optimization problem. In [3], all caregivers' routes must start and end at their residences and the working time is of type 1. The districting step significantly reduces the problem size, leading to notable improvements in computational efficiency while maintaining near-optimal total cost in terms of hiring and transportation costs for caregivers.

In [1], the authors propose a bi-objective model for a daily HHC routing and scheduling problem, aiming to minimize both transportation cost and the number of vehicles used. In their model, the hiring cost for caregivers is not considered since the number of caregivers is predetermined. Each caregiver's route starts and ends at a depot, so the working time is of type 1. Skill level requirements are not considered in their model and the service time for all requests is fixed at 20 minutes. The authors develop two evolutionary algorithm (EA) approaches, both employing a pareto selection scheme.

In [6], the authors define a Markov decision process (MDP) for the HHC problem and they build up a Q-learning model, a model-free reinforcement learn-

ing (RL) algorithm, on top of the MDP. The study in [6] employs "mixed time window constraints", which prohibit caregivers to arrive late but allow them to arrive early and wait until the patients' time windows begin. In [6], all caregivers' routes must start and end at the depot, so the working time is of type 1. The proposed Q-learning model effectively balances the trade-off between longer allowable waiting times — offering greater scheduling flexibility — and the inefficiency associated with excessive idle time for caregivers.

In [5], route construction relies heavily on insertion-based scheduling, where patients are sequentially inserted into caregivers' schedules. Since the duration of requests may not perfectly align with the available time slots in caregivers' schedules, gaps may occur between assignments. Thus, caregivers might experience idle time or early arrival for their next assignment. In [5], additional requests are allowed to be submitted during serving, and the HHC system will then on decide, in real-time, whether to accept the requests or not. Reference [5] assumes multiple depots, and for each caregiver, the route must start and end in respective depot. The working time is of type 1. The objective of [5] is to maximize the number of patients served, and it is shown to be able to handle real-time requests submitted by patients during the bi-weekly schedule.

Table 1. Summary of related works for home health care. "(1)" means "(some caregivers may not be hired)", "(2)" means "(all caregivers must be hired)", "Time Window" means "Time Window Constraints", "min." means "minimize", and "max." means "maximize".

References	Set of caregivers	Span	Endpoints	Time Window	Objective
2021 [3]	given in advance (1)	1 week	type 1	hard	min. hiring and transportation costs
2022 [1]	given in advance (2)	1 day	type 1	no	min. vehicles and transportation costs
2023 [6]	given in advance (2)	1 day	type 1	mixed	min. travel and wait times
2024 [5]	given in advance (2)	2 weeks	type 2	mixed	max. number of patients served
this manuscript	adjusted	1 day	type 2	hard	min. hiring and transportation costs

This study is motivated by the following observations.

O1. In [1, 3, 5, 6], the sets of caregivers are given in advance. Thus, there might have redundant caregivers or the number of caregivers might be insufficient.

O2. In [3], the districting approach imposes a rigid structure in which caregivers are restricted to operate solely within their assigned districts, thereby excluding potentially better solutions that span across district boundaries.

O3. None of [3, 5, 6] asks a caregiver to finish the patient's request within the given time window. In other words, [3, 5, 6] may not be patient-friendly since a patient may receive a service outside the expected time window.

O4. In [5, 6], a caregiver is allowed to arrive at the patient's residence "earlier than" the given time window and wait until the time window begins, causing excessive waiting and idle time in the caregiver's schedule. For example, instance C in Figure 5 [6] shows that among 480 minutes of working time, the average waiting time is more than 160 minutes.

In this study, we aim to minimize the cost caused by caregivers' hiring and transportation. Accordingly, the set of caregivers is adjusted (i.e., not given in advanced). The span considered is 1 day, as we believe that it is not difficult to

extend our solution to other spans. We consider working time of type 2 since, in our home country, a caregiver's working time begins with the first service and ends with the final service. The time windows are assumed to be hard constraints to ensure patient-friendliness, as defined in Subsection 2.2.

The main contributions of this study are:

- We discard the usage of a predefined caregiver set, which is used in [1, 3, 5, 6]. Instead, we adjust the number of caregivers needed, thus finding a suitable number of caregivers.
- We discard the concept of districting in [3]. Instead, we propose an algorithm, called hiCH, which is based on a hierarchical convex hull approach to minimize both the hiring and the transportation costs of caregivers. To the best of our knowledge, we are the first to adopt convex hull to solve HHC problem.
- We sort the requests based on their levels and time windows, and combine this with hiCH. This results in a new algorithm: D-hiCH.
- We further enhance the solution by combining genetic algorithm (GA) with D-hiCH. By utilizing the solution obtained from D-hiCH as an initial solution, we can accelerate the convergence of GA and search for solution that outperforms D-hiCH.

2 Time Window, Patient-Friendliness, and Problem Formulation

In this section, we first provide the background on time windows. Then, we introduce a new constraint for HHC, aimed at achieving patient-friendliness. Next, we describe our system assumptions. Finally, we formally define the HHC problem addressed in this study.

The following notations are needed. Let $[e_i, l_i]$ denote a time window, where e_i and l_i are the earliest and latest service time requested by patient i, respectively. Additionally, let a_i, b_i, and f_i denote the arrival time, the service beginning time, and the service finishing time of the caregiver assigned to patient i, respectively.

2.1 Time Window Constraint

There are four types of time windows: hard time window (HTW), soft time window (STW), mixed time window (MTW), and hard & soft time window (HTW & STW). In this study, we only consider HTW, in which a caregiver is only allowed to arrive within the time window, i.e., HTW requires that $a_i \in [e_i, l_i]\ \forall i$ and it prohibits both early arrival and tardiness. An illustration of HTW can be found in Figure 2. See [6] for details of other types of time windows.

2.2 Patient-Friendliness Constraint

Let n denote the number of patients. In this manuscript, we assume that all requests are fulfilled under HTW constraints. In order to better respect patient preferences, we introduce the following patient-friendliness constraint:

For each i, $1 \leq i \leq n$, patient i must have $e_i \leq f_i \leq l_i$.

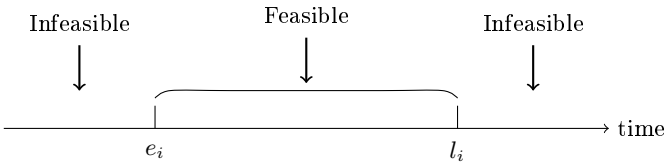


Fig. 2. Hard time window (HTW): arrows indicate caregivers' arrivals.

This constraint enforces that not only must the service begin within the time window, but the completion of service must also fall within it. That is, the entire service must be provided during the patient's available time window.

2.3 System Assumptions

The following system assumptions are used. The span of the considered HHC problem is 1 day. Each patient submits exactly one request. Each patient must be served once and by the same caregiver. HTW is used. Each caregiver can only provide services to patients with required skill levels lower than or equal to the caregiver's skill level. Each caregiver's workload upper bound is $\mathcal{W}$. (In our simulations, $\mathcal{W}$ is 480 minutes.) The service time required by each patient is strictly smaller than $\mathcal{W}$. Each caregiver's working time is of type 2, i.e., the working time begins from starting serving the first patient and ends after finishing serving the last patient. Each caregiver's transportation cost is proportional to the distance traveled.

2.4 Notations

Since the span of the considered HHC problem is 1 day and each patient submits exactly one request, we will not distinguish between patients and their requests. The following notations are introduced for patients. The HHC provider will receive a list of requests submitted by the set of patients $\mathcal{P} = \{1, 2, \ldots, n\}$. The request of patient i, $1 \leq i \leq n$, is represented as a 4-tuple (e_i, l_i, q_i^p, s_i), where $[e_i, l_i]$ is the time window (e_i is the earliest time the service may begin, whereas l_i is the latest time the service must be completed), q_i^p is the qualification level needed to perform the service, and s_i is the service time required. To ensure feasibility, each request (e_i, l_i, q_i^p, s_i) must satisfy $s_i \leq l_i - e_i$. Also, by our assumptions, $s_i < \mathcal{W}$ holds for each patient i, $1 \leq i \leq n$.

The following notations are introduced for caregivers. Let $\mathcal{C}$ denote the set of caregivers. Let $m = |\mathcal{C}|$ be the number of caregivers. Initially, $C = \emptyset$ and $m = 0$. Each caregiver $j \in \mathcal{C}$ is associated with a skill level q_j^c. Notice that a caregiver is allowed to serve for requests with skill levels lower than or equal to her/his qualification level. Let r_j denote the route of visits for caregiver j, let n_j denote the number of patients in r_j (i.e., number of patients assigned to caregiver j), and let r_j^i denote the i-th visited patient on r_j. Let h_j and w_j denote the hiring cost and the working time of caregiver j, respectively.

We model this spatial relationship between all patients' residences by constructing a graph $G = (V, E)$, where V consists of nodes representing residences of patients and E is the set of edges weighted by the distance $d_{u,v}$ between any two patients $u, v \in V$. By convention, G is a complete graph.

2.5 Problem Formulation

The objective of our HHC problem is to minimize the cost incurred by hiring caregivers and transportation between patients' residences for caregivers:

$$\min \quad \underbrace{\sum_{j \in \mathcal{C}} h_j}_{\text{hiring cost}} \quad + \quad \underbrace{\sum_{j \in \mathcal{C}} \sum_{i=1}^{n_j - 1} tr \times d_{r_j^i, r_j^{i+1}}}_{\text{transportation cost}} \tag{1}$$

$$\text{s.t.:} \quad q^p_{r_j^i} \leq q^c_j, \quad \forall\, j \in \mathcal{C},\ \forall\, 1 \leq i \leq n_j \tag{2}$$

$$e_i \leq b_i \leq f_i \leq l_i, \quad \forall\, i \in \mathcal{P} \tag{3}$$

$$\sum_{i=1}^{n_j - 1} d_{r_j^i, r_j^{i+1}} \ + \ \sum_{i=1}^{n_j} s_{r_j^i} \ \leq \ \mathcal{W}, \quad \forall\, j \in \mathcal{C} \tag{4}$$

Notice that in our assumption, transportation cost incurred by each caregiver will be proportional to the distance travelled. Thus, the transportation cost incurred from travelling from u to v would be $tr \times d_{u,v}$, where tr is the transportation cost per unit of time travelled. We now explain (1)–(4).

The objective (1) minimizes the sum of hiring cost and transportation cost. Constraint (2) ensures that all patient requests are assigned to caregivers with the appropriate skill levels. Constraint (3) guarantees that all services are provided within the time windows specified by the patients, that is, the patient-friendliness is guaranteed. Finally, constraint (4) ensures that the total working time — including both travelling and service time — for each caregiver does not exceed the working time upper bound.

3 Our Solution to HHC: hiCH

We first give the definition of a convex hull. The *convex hull* of a set of ≥ 3 points is the smallest convex polygon containing all points in the set. Idea of our algorithm hiCH is as follows.

Suppose that j caregivers have already been hired and let $\mathcal{P}$ be the set of yet unvisited patients. Set $j = j + 1$ and let $\mathcal{H}_j$ be the convex hull of points corresponding to the residences of patients in $\mathcal{P}$. Among patients on $\mathcal{H}_j$, we find a patient i_1 with the highest required skill level $q^p_{i_1}$ under

$$i_1 = \underset{i \in \mathcal{H}_j}{\operatorname{argmax}}\, q^p_i,$$

and hire a caregiver c_j with the corresponding level $q^c_{c_j} = q^p_{i_1}$ to provide the service. In general, if caregiver c_j is serving with current schedule being $r_j = [\, i_1, i_2, \ldots, i_k \,]$, then the $(k+1)$-th patient is selected from the pool $\mathcal{S}^k_j$ below.

Let

$$\mathcal{S}^k_j = \{i \in \mathcal{P} \mid \underbrace{q^p_i \le q^c_{c_j}}_{\text{(i)}},\ \underbrace{e_i \le f_{i_k} + d_{i_k,i}}_{\text{(ii)}},\ \underbrace{f_{i_k} + d_{i_k,i} + s_i \le l_i}_{\text{(iii)}},\ \text{(iv)}\} \tag{5}$$

where (iv) is

$$\sum_{\ell=1}^{k} s_{i_\ell} + \sum_{\ell=1}^{k-1} d_{i_\ell,i_{\ell+1}} + d_{i_k,i} + s_i \le \mathcal{W}.$$

Patient i_{k+1} is then selected by

$$i_{k+1} = \operatorname*{argmin}_{i \in \mathcal{S}^k_j} d_{i_k,i}$$

and added to route r_j. The route is eventually completed if we cannot find any new candidate from the pool.

Input to our algorithm hiCH are the set of requests submitted by patients and a matrix containing distances between each pair of patients. Output of our algorithm hiCH are the set of hired caregivers and the routes (i.e., schedules) of caregivers. We now are ready to list our algorithm hiCH in Algorithm 1.

We use the residences of 30 patients whose data are from dataset Ins 1-30-2-s1 in [3] to illustrate how our hiCH successively finds convex hulls and arranges routes for caregivers. See Figure 3 for an illustration. In this figure, the following conventions are adopted.

Nodes colored red are the residences of patients whose caregivers' qualification levels are level 3 (the highest). Nodes colored blue are the residences of patients whose caregivers' qualification levels are level 2 (the median). Nodes colored green are the residences of patients whose caregivers' qualification levels are level 1 (the lowest). Polygons shown by solid lines are the convex hulls. Paths shown by solid lines are the routes of caregivers. In the route for a caregiver, "start" means the "starting residence" and "end" means the "ending residence" of the route. After each route is formed, the visited patients will be marked in grey to indicate that their requests have been fulfilled. This process continues until all patients' requests have been fulfilled.

We summarize the outcome of hiCH for dataset Ins 1-30-2-s1 in [3] in Table 2.

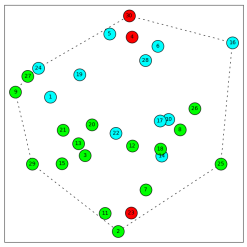
Algorithm 1: hiCH

Input: (1) $\mathcal{P}$: the set of requests submitted by patients,
(2) $(d_{i,j})$: a matrix containing distances between each pair of patients.
Output: (1) $\mathcal{C}$: the set of hired caregivers,
(2) $r_1, r_2, \ldots, r_{|\mathcal{C}|}$: the routes (i.e., schedules) of caregivers.

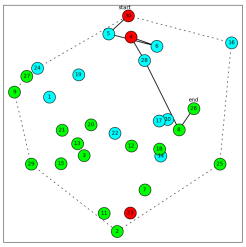
$\mathcal{C} \leftarrow \emptyset$; $j \leftarrow 0$
while $\mathcal{P} \neq \emptyset$ **do**
  $\mathcal{H} \leftarrow$ convex hull of $\mathcal{P}$
  $i \leftarrow \text{argmax}_{k \in \mathcal{H}}\, q_k^p$
  $j \leftarrow j + 1$
  # hire a new caregiver
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{c_j\}$, where c_j is a caregiver with skill level $q_j^c = q_i^p$
  $r_j \leftarrow [\,]$ # r_j: route of c_j
  $w_j \leftarrow 0$ # w_j: workload of c_j
  # arrange the route for this new caregiver
  $r_j \leftarrow r_j + [\,i\,]$ # add patient i to route of c_j
  $w_j \leftarrow w_j + s_i$ # provide service to patient i
  $\mathcal{P} \leftarrow \mathcal{P} \setminus \{i\}$ # delete patient i
  poolEmpty $\leftarrow$ False
  while $w_j \leq \mathcal{W}$ **and** NOT poolEmpty **do**
    form pool $\mathcal{S}$ by using (5)
    if $\mathcal{S} \neq \emptyset$ **then**
      $i' \leftarrow \text{argmin}_{i \in \mathcal{S}}\, d_{i,i'}$ # choose the next patient i'
      $r_j \leftarrow r_j + [\,i'\,]$ # add patient i' to route of c_j
      $w_j \leftarrow w_j + d_{i,i'}$ # travel from patient i to patient i'
      $w_j \leftarrow w_j + s_{i'}$ # provide service to patient i'
      $\mathcal{P} \leftarrow \mathcal{P} \setminus \{i'\}$ # delete patient i'
      $i \leftarrow i'$
    else
      poolEmpty $\leftarrow$ True
return $\mathcal{C}$ and $r_1, r_2, \ldots, r_{|\mathcal{C}|}$.

Table 2. Outcome of hiCH for dataset Ins 1-30-2-s1 in [3].

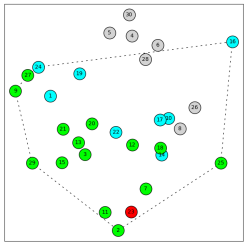
Caregiver j	Skill level	Working time	Service time	Traveling time	Route of the caregiver
$j = 1$	3	453	401	52	$r_1 = [\,30, 5, 6, 4, 28, 8, 26\,]$
$j = 2$	2	433	394	39	$r_2 = [\,16, 10, 14, 17, 12\,]$
$j = 3$	2	463	421	42	$r_3 = [\,19, 27, 9, 1, 13\,]$
$j = 4$	2	455	415	40	$r_4 = [\,24, 21, 20, 3, 22\,]$
$j = 5$	1	473	383	90	$r_5 = [\,29, 15, 11, 2, 25\,]$
$j = 6$	3	299	267	32	$r_6 = [\,23, 7, 18\,]$
Total	13	2576	2281	295	



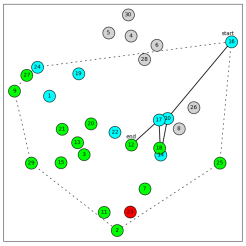
(a) Convex hull $\mathcal{H}_1$.



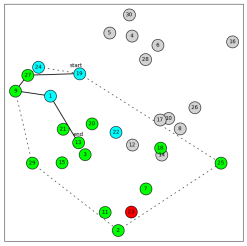
(b) Caregiver 1's route.



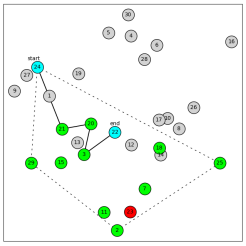
(c) Convex hull $\mathcal{H}_2$.



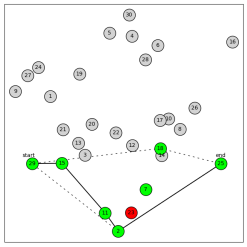
(d) Caregiver 2's route.



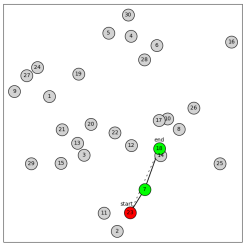
(e) Caregiver 3's route.



(f) Caregiver 4's route.



(g) Caregiver 5's route.



(h) Caregiver 6's route.

Fig. 3. Illustration of the execution of hiCH.

4 Improved Version of hiCH: D-hiCH

We also implement another version of hiCH, called D-hiCH, such that D-hiCH sorts patients' requests in decreasing order of required skill level and after sorting, D-hiCH executes like hiCH. We omit the pseudocode of D-hiCH.

5 GA Version of hiCH: GA + D-hiCH

In [1], the authors demonstrate that evolutionary algorithms (EA) can effectively solve HHC routing and scheduling problems by searching diverse solution spaces while handling complex constraints. Their findings highlight the potential of population-based search methods for HHC, providing the motivation of incorporating GA into our work as an additional refinement tool. Thus, we try to further improve the solution obtained from D-hiCH by integrating a Genetic Algorithm (GA) to explore potentially better assignments and routing structures.

We find that applying GA alone makes it difficult to converge effectively within a reasonable number of generations, and we believe this difficulty arises from the immense size of the search space ($n!$), where n is the number of patients. We also find that if the initial population is fully randomized, as in the conventional GA setup, the algorithm requires a significant amount of time before high-quality individuals appear. To address the aforementioned issues, we replace a portion of the initial population with solutions produced by D-hiCH under minor perturbations. Although our strategy slightly narrows the search space, it injects individuals of relatively high fitness into the population, thereby enabling faster convergence. We refer to this approach as the *search-space-narrowing strategy*. The performance of our search-space-narrowing strategy (or, equivalently, GA + D-hiCH) will be demonstrated in the final table of this manuscript; details of the strategy are described below.

We first encode the HHC problem into chromosomes so that GA can be applied. Given a list of patients, we treat each patient as an index from $\{1, \ldots, n\}$. Thus, each chromosome is represented as a permutation of n elements, which is composed of genes as patients. Assignments and routing are then constructed by scanning the chromosome from left to right: a caregiver is assigned patients sequentially until no more patients can feasibly added to their schedule. The resulting subsequence becomes the caregiver's visiting sequence, and the skill level of the caregiver is the highest required skill level among the patients in the subsequence. The process repeats until all patients have been assigned. The fitness of an individual is the total cost obtained from the entire sequence, namely the sum of caregiver hiring cost and transportation cost for each subsequence.

We find that applying GA alone is difficult to converge effectively. We also find that if the initial population is fully randomized, the algorithm requires significant amount of time before high-quality individuals appear. To address these issues, we thus replace a portion of the initial population with the solution produced from D-hiCH under minor perturbations. In particular, we apply three types of perturbations to the D-hiCH solution:

- Swap — exchange two genes.

– Insert — remove a gene and insert it at a random position.
– Shuffle — randomly permute a selected subsequence.

Each perturbation is applied multiple times, and approximately 25% of the population is generated from each perturbation type. The remaining 25% of the population remains fully random to preserve diversity.

6 Simulation Setup and Results

Our D-hiCH is implemented in Python. Both GA and GA + D-hiCH are implemented in Python using pyGAD. All of our algorithms are executed on a personal computer equipped with an Intel(R) Core(TM) Ultra 7 265KF CPU and 64 GB RAM. We categorize the datasets into two groups based on problem size. For small instances, we use the datasets generated from [3]. For larger instances, we use the real-world datasets provided by the Austrian Red Cross, originally introduced in [2].

– Small instances: $n = 10, 20, 30, 40$.
– Large instances: $n = 75, 100, 125$.

We now give the details of hiring cost, transportation cost, tr, and workload upper bound. First, the hiring cost of a caregiver is based on the caregiver's skill level and we assume: the hiring cost of a caregiver of level 1 is 1000 units, the hiring cost of a caregiver of level 2 is 1.2 times that of a caregiver of level 1, and the hiring cost of a caregiver of level 3 is 1.2 times that of a caregiver of level 2. The transportation cost incurred from travelling from u to v is set to $tr \times d_{u,v}$, where tr is the transportation cost per unit of time travelled. In our simulations, tr is set to 1 unit. The workload upper bound $\mathcal{W}$ for each caregiver is set to 480 units, which includes both the service and the transportation. The value 480 comes from the calculation $8 \times 60 = 480$ if each caregiver works at most 8 hours a day. We now list parameters used in our simulations in Table 3.

Table 3. Parameters and setting.

Parameter	Value
n	a value in $\{10, 20, 30, 40, 75, 100, 125\}$
Hiring cost of a caregiver of level 1	1000 units
Hiring cost of a caregiver of level 2	1200 units
Hiring cost of a caregiver of level 3	1440 units
tr	1 unit
$\mathcal{W}$	480 units

Each group adopts the same GA configurations, described as follows: Parent selection is performed using tournament selection, where a subset of individuals is randomly chosen and the one with the highest fitness is selected as a parent. The size of this subset is determined by the selection number. Selected parents then undergo crossover to produce offspring. We apply order crossover[3], which is an

[3] https://stackoverflow.com/questions/26518393/order-crossover-ox-genetic-algorithm

approach well-suited for permutation-based representations. Mutations may also occur under a predefined mutation probability, using swap mutation, where two randomly chosen genes exchange their positions. Finally, because the population size must remain fixed across generations, only a portion of individuals from the combined pool of parents and offspring are retained. The replacement parameter specifies how many individuals from the current generation survive into the next. Each instance is executed 30 times, and the run containing the individual with the best fitness is selected as the final result. Table 4 summarizes the parameters.

Table 4. Simulation setup for two groups of dataset.

Parameter	$n = 10, 20, 30, 40$	$n = 75, 100, 125$
Population	200	500
Generation	100	300
Selection	tournament	tournament
Selection number	2	5
Crossover	order	order
Mutation	swap	swap
Mutation probability	0.3	0.5
Replacement	50	100

We compare D-hiCH, GA, and GA + D-hiCH. Due to space limitation, we omit the outcome of small instances. Outcome of large instances is shown in Table 5. The followings (i)–(iii) can be observed from Table 5. (i) As n grows larger, GA becomes less effective due to the exponential growth of the search space. (ii) The GA + D-hiCH approach, however, maintains good performance. (iii) Although incorporating high-quality individuals derived from D-hiCH narrows the search space, these individuals help guide the algorithm toward solutions that remain competitive among the three algorithms.

7 Concluding Remarks

In this manuscript, we consider the HHC problem with the goal of minimizing both the total hiring cost and the total transportation cost of caregivers, while maintaining patient-friendliness. We utilize a hierarchical convex hull approach to find a suitable number of caregivers. Based on this approach, we propose three algorithms: hiCH, D-hiCH, and GA + D-hiCH. We conduct extensive simulations on real-world datasets. Simulation results show that GA + D-hiCH is a promising solution for the HHC problem. As a final remark, a preliminary version of this manuscript was in [4].

References

1. M. Belhor, A. El-Amraoui, A. Jemai, and F. Delmotte, "Multiobjective evolutionary algorithm for home health care routing and scheduling problem," in: Proceedings of 2022 8th International Conference on Control, Decision and Information Technologies (CoDIT), Istanbul, Turkey, pp. 1592-1596, 2022.

Table 5. Comparisons of D-hiCH, GA, and GA + D-hiCH with respect to 75, 100, and 125 patients: "n" means "number of patients", each four-tuple (x, y, z, w) represents (hiring cost, transportation cost, total cost, number of caregivers hired), and "†" means "the optimal one".

Dataset in [2]	n	D-hiCH	GA	GA + D-hiCH
U1-n75	75	(17360, 678, 18038, 14)	(15520, 791, 16351, 13)	(15720, 719, 16439, 13)
U2-n75	75	(15920, 716, 16636, 14)	(14680, 834, 15514, 13)	(13480, 642, 14122, 12)
U3-n75	75	(18960, 784, 19744, 16)	(14480, 1043, 15523, 13)	(14720, 759, 15479, 13)
U4-n75	75	(17000, 616, 17616, 14)	(14680, 794, 15474, 13)	(13920, 653, 14573, 12)
U5-n75	75	(19240, 680, 19920, 16)	(15160, 857, 16017, 13)	(14400, 729, 15129, 12)
S1-n75	75	(17800, 1246, 19046, 14)	(17520, 1717, 19237, 15)	(16600, 1158, 17758, 13)
S2-n75	75	(16560, 1342, 17902, 14)	(17080, 1517, 18597, 15)	(15880, 1408, 17288, 14)
S3-n75	75	(19200, 1282, 20482, 16)	(16480, 1561, 18041, 15)	(15680, 1304, 16984, 14)
S4-n75	75	(17520, 1352, 18872, 15)	(18120, 1580, 19700, 16)	(14880, 1332, 16212, 13)
S5-n75	75	(18440, 1320, 19760, 15)	(17560, 1426, 18986, 15)	(15360, 1331, 16691, 13)
Average		(17800, 1001.6†, 18801.6, 14.8)	(16128, 1212, 17344, 14.1)	(15064†, 1003.5, 16067.5†, 12.9†)
U1-n100	100	(25320, 768, 26088, 21)	(21200, 1387, 22587, 18)	(21000, 798, 21798, 18)
U2-n100	100	(22600, 894, 23494, 19)	(21760, 1265, 23025, 19)	(18800, 853, 19653, 16)
U3-n100	100	(19960, 780, 20740, 17)	(20520, 1125, 21645, 18)	(18320, 891, 19211, 16)
U4-n100	100	(21600, 794, 22394, 18)	(20720, 1385, 22105, 18)	(19120, 804, 19924, 17)
U5-n100	100	(25680, 812, 26492, 21)	(23720, 1254, 24974, 21)	(21560, 949, 22509, 19)
S1-n100	100	(28120, 1544, 29664, 23)	(24200, 2258, 26458, 21)	(23640, 1664, 25304, 20)
S2-n100	100	(27000, 1488, 28488, 23)	(23720, 2120, 25840, 21)	(23360, 1336, 24696, 20)
S3-n100	100	(23280, 1660, 24940, 19)	(23960, 2235, 26195, 21)	(20000, 1606, 21606, 17)
S4-n100	100	(23360, 1768, 25128, 20)	(23320, 2258, 25578, 21)	(20880, 1731, 22611, 19)
S5-n100	100	(29880, 1446, 31326, 25)	(26840, 2073, 28913, 23)	(25280, 1579, 26859, 21)
Average		(24680, 1195.4†, 25875.4, 20.6)	(22996, 1736, 24732, 20.1)	(21196†, 1221.1, 22417.1†, 18.3†)
U1-n125	125	(29880, 998, 30878, 25)	(27320, 1830, 29150, 24)	(24560, 1049, 25609, 21)
U2-n125	125	(25400, 996, 26396, 22)	(26800, 1779, 28579, 24)	(23000, 953, 23953, 20)
U3-n125	125	(27760, 1062, 28822, 24)	(26400, 1611, 28011, 23)	(23720, 1050, 24770, 21)
U4-n125	125	(29640, 894, 30534, 25)	(26200, 1739, 27939, 23)	(25160, 952, 26112, 22)
U5-n125	125	(29040, 896, 29936, 24)	(26360, 1727, 28087, 23)	(24520, 1048, 25568, 21)
S1-n125	125	(28640, 1812, 30452, 24)	(30600, 2614, 33214, 27)	(27000, 1756, 28756, 23)
S2-n125	125	(29880, 1924, 31804, 25)	(30880, 2814, 33694, 27)	(27080, 1937, 29017, 23)
S3-n125	125	(30560, 1752, 32312, 25)	(30200, 3081, 33281, 27)	(26320, 1941, 28261, 22)
S4-n125	125	(31200, 1704, 32904, 27)	(30360, 2886, 33246, 27)	(27360, 1866, 29226, 24)
S5-n125	125	(29600, 1814, 31414, 25)	(29800, 2960, 32960, 26)	(26960, 1759, 28719, 23)
Average		(29160, 1385.2†, 30545.2, 24.6)	(28492, 2304.1, 30816.1, 25.1)	(25568†, 1431.1, 26999.1†, 22†)

2. C. Fikar and P. Hirsch, "A matheuristic for routing real-world home service transport systems facilitating walking," Journal of Cleaner Production, vol. 105, pp. 300-310, 2015.
3. E. Nikzad, M. Bashiri, and B. Abbasi, "A matheuristic algorithm for stochastic home health care planning," European Journal of Operational Research, vol. 288, pp. 753-774, 2021.
4. S. Su, *A Novel Algorithm for Assignment and Routing in the Home Health Care Problem*, (Master thesis), National Yang Ming Chiao Tung University, Hsinchu, Taiwan, 2025.
5. Q. Ta-Dinh, T.-S. Pham, M.H. Hà, and L.-M. Rousseau, "A reinforcement learning approach for the online dynamic home health care scheduling problem," Health Care Management Science, vol. 27, pp. 650–664, 2024.
6. T. Zhang, Y. Liu, X. Yang, J. Chen, and J. Huang, "Home health care routing and scheduling in densely populated communities considering complex human behaviours," Computers and Industrial Engineering, vol. 182, 109332, 2023.

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Instance Space Analysis for Modular High School Timetabling*

Andreas Krystallidis[1,2][0009−0007−0183−3038], Rubén Ruiz-Torrubiano[1][0000−0001−9314−0739], and Nysret Musliu[2][0000−0001−9314−0739]

[1] IMC Krems University of Applied Sciences, Piaristengasse 1, Krems, 3500, Austria
ruben.ruiz@imc.ac.at

[2] TU Wien, Institute for Logic and Computation, Favoritenstraße 9-11, Vienna, 1040, Austria
andreas.krystallidis@tuwien.ac.at
nysret.musliu@tuwien.ac.at

Abstract. High school timetabling is a challenging combinatorial optimization problem involving numerous hard and soft constraints and competing stakeholder interests. In modular high schools, student course selections introduce additional complexity beyond standard time and resource constraints. Structural and regulatory differences between schools further increase instance variability, making it unlikely that a single solution approach performs best across all cases.

To address this variability, identifying instance characteristics that explain differences in algorithm performance is essential for informed algorithm selection. In this paper, we apply Instance Space Analysis (ISA) to the modular extension of the Extended High School Timetabling format (XHSTT). We analyze two fundamentally different solution approaches: Simulated Annealing (SA), and an Adaptive Large Neighborhood Search (ALNS) method.

Our study considers approximately 1500 diverse instances, including real-world modular XHSTT instances, existing instances extended with student course selections, and purely artificial modular instances. The modified and artificial instances represent novel contributions to the modular XHSTT benchmark set.

Using ISA, we characterize the instance space and evaluate algorithm performance. The results provide insights into structural properties influencing solver behavior and demonstrate ISA's potential for supporting algorithm selection in modular high school timetabling.

Keywords: Educational Timetabling · High School Timetabling · Instance Space Analysis · Algorithm Selection

1 Introduction

Creating high school timetables is a complex combinatorial optimization problem that requires balancing the interests of multiple stakeholders while satisfying

* A. Krystallidis and R. Ruiz-Torrubiano acknowledge financial support from the Research Promotion Agency of Lower Austria (GFF) under project grant FTI21-A-002.

numerous hard and soft constraints. In modular high schools, this challenge is further complicated by student course selections, in addition to the standard constraints related to time and resource allocation. Moreover, no two schools are identical: they differ in the number and composition of teachers, students, and rooms, as well as in course structures and institutional or legal requirements that vary across countries and school types.

Significant effort has been made within the scheduling community to develop unified formats capable of capturing the diverse requirements of real-world timetabling problems. The most widely used format for high school timetabling is the Extended High School Timetabling format (XHSTT) [6]. While this format is highly expressive, it assumes relatively rigid student curricula and does not naturally support student course selections. To address this limitation, the modular extension of XHSTT, referred to as *modular XHSTT* [3], was introduced. Although initial solution approaches for this extended format have been proposed [4], the availability of benchmark instances remains limited. Furthermore, for both XHSTT and modular XHSTT, there is still a lack of systematic analysis of how instance characteristics influence the performance of automated scheduling methods.

This paper aims to address these gaps through the following contributions:

- We present two tools for generating modular XHSTT instances: an *Instance Creator*, which produces artificial instances from scratch, and an *Instance Extender*, which augments existing XHSTT instances with modular components to generate more realistic scenarios.
- We develop an Adaptive Large Neighborhood Search (ALNS) approach for solving the modular XHSTT problem.
- We identify and extract a comprehensive set of features that characterize modular XHSTT instances.
- We apply the Instance Space Analysis (ISA) framework to visualize and analyze the relationship between instance features and algorithm performance, and to examine the distribution of instances from different sources.
- We compare the proposed ALNS approach with a Simulated Annealing (SA) heuristic [4], and identify regions of the instance space in which each method performs favorably.

The remainder of the paper is structured as follows. Section 2 introduces the modular XHSTT problem. Section 3 describes the ISA framework. In Section 4, we detail the data generation process, including instance generation, algorithm configuration, and feature extraction. Section 5 presents the results of the analysis. Finally, Section 6 concludes the paper and outlines directions for future work.

2 Modular XHSTT

The modular Extended High School Timetabling (modular XHSTT) format [3] is an extension of the XHSTT model designed to represent school systems in which students select part of their curriculum individually.

2.1 Base XHSTT Model

In the standard XHSTT format, a timetable consists of a set of events that must be assigned to discrete time slots organised into days and, optionally, time groups (e.g., mornings or afternoons). Events may be split into multiple sub-events subject to structural restrictions. Each event is associated with event resources, representing either fixed assignments (e.g., a specific teacher) or requirements for certain resource types.

The model provides 16 constraint types that regulate time assignments, resource allocations, and structural properties of the timetable. These include:

- assignment constraints ensuring that events receive appropriate times and resources,
- event splitting constraints controlling the number and duration of sub-events,
- preference constraints penalizing undesirable times or resources,
- conflict and availability constraints preventing clashes and assignments at unavailable times,
- temporal relationship constraints such as linking or ordering events,
- workload and load-balancing constraints limiting total work, idle times, or daily activity.

Each constraint is defined by a deviation-based cost function, a weight, and a hardness flag. The objective value of a timetable is the weighted sum of all constraint violations. The flexibility of grouping arbitrary sets of events, times, and resources makes the format expressive enough to model a wide range of institutional requirements.

2.2 Extensions for Modular Schools

While the original XHSTT specification captures many structural aspects of school timetabling, it does not model student course selection. Modular XHSTT therefore introduces additional mechanisms to represent elective structures and the resulting class formation requirements.

- **StudentChoiceConstraints** model elective behavior by specifying, for each student resource, sets of event groups from which a minimum and maximum number must be attended. These constraints allow flexible modeling of alternative courses, option blocks, and nested choice structures.
- **BalanceClassSizeConstraints** address the distribution of students across parallel events. The constraint limits the maximum difference in the number of assigned student resources between event groups. This supports balanced class formation when multiple equivalent sections of a course are offered.

Together, these extensions allow modular XHSTT to represent elective systems while maintaining compatibility with the underlying XHSTT structure and evaluation mechanism.

3 Framework: Instance Space Analysis and Algorithm Selection

ISA provides a framework for analyzing the relationship between structural properties of problem instances and algorithm performance. The central idea is to represent instances in a low-dimensional space, typically two-dimensional, such that similarities in instance characteristics and solver behavior become visible.

The approach builds on the algorithm selection framework of Rice [7] and was formalized by Smith-Miles et al. [8]. By jointly analyzing instance features and performance data, ISA identifies regions in which specific algorithms perform well or poorly, revealing their strengths and complementarities. This provides more detailed insight than aggregate performance measures over heterogeneous benchmark sets.

In addition, ISA supports the evaluation of benchmark quality by enabling the analysis of instance diversity, the detection of biases, and the identification of underrepresented regions of the instance space.

The overall framework is illustrated in Figure 1. Let $\mathcal{P}$ denote the space of all possible problem instances (here, modular XHSTT instances), and $\mathcal{I} \subseteq \mathcal{P}$ the finite set of instances considered. For each instance $x \in \mathcal{I}$, a set of numerical characteristics is extracted and represented as a feature vector $f(x) \in \mathcal{F}$, defining the feature space $\mathcal{F}$. A portfolio of algorithms $\mathcal{A}$ is available, where an individual algorithm is denoted by $\alpha \in \mathcal{A}$. Applying these algorithms to the instances yields performance metrics $y(\alpha, x)$ (measured here by objective value), which define the performance space $\mathcal{Y}$.

While the original framework [7] aims to predict algorithm performance from this meta-data, ISA instead projects the feature space into a two-dimensional instance space that preserves relevant structure. The directional flow in Figure 1 reflects a two-part process rather than a closed operational loop: first, the feature space $\mathcal{F}$ and performance space $\mathcal{Y}$ are jointly utilized via dimension reduction to construct the underlying instance space. Once this space is established, the algorithm selection model uses an instance's coordinates to recommend a specific algorithm α from the algorithm space $\mathcal{A}$, the actual or predicted performance of which is ultimately represented within $\mathcal{Y}$.

The resulting instance space supports: (i) identification of uncovered regions, (ii) assessment of feature quality, (iii) characterization of algorithm strengths and weaknesses, (iv) feature-based algorithm selection, (v) detailed algorithm comparison, and (vi) analysis of instance distribution and potential imbalances.

4 Methodology

In this section, we discuss how the meta-data of this study is generated for the modular XHSTT problem.

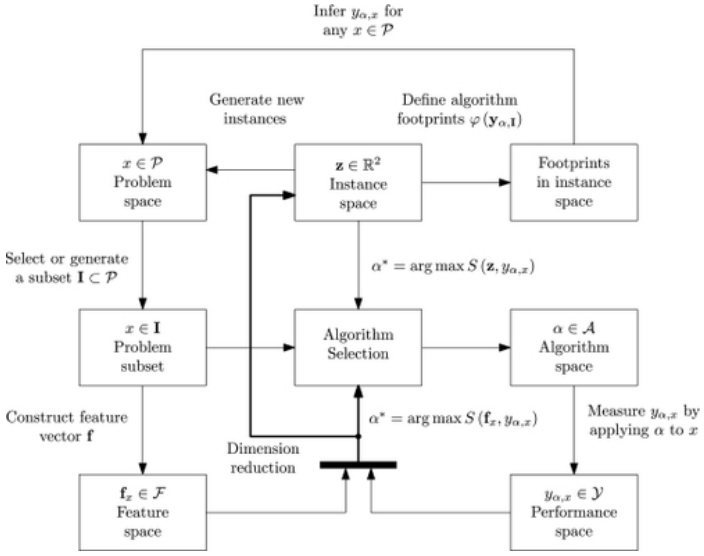


Fig. 1: Instance Space Analysis Framework by Smith-Miles et al. [8] extending the Algorithm Selection Framework by Rice [7]

.

4.1 Problem Subset $\mathcal{I}$

Publicly available modular XHSTT instances are currently limited to 18 real-world German high school instances [3, 4], which is insufficient for ISA and algorithm selection due to the need for larger and more diverse benchmark sets.

To address this, we introduce two complementary approaches (Figure 2). The *Instance Creator* generates fully synthetic modular XHSTT instances, focusing on course selection structures, while the *Instance Extender* augments existing XHSTT instances by introducing artificial course selections.

Our final data set consists of three subsets: 18 real-world German instances (**Germany**), 999 synthetic instances (**Artificial**), and 520 extended instances based on real-world data (**modXHSTT**). The problem subset is defined as $\mathcal{I} = \textbf{Germany} \cup \textbf{Artificial} \cup \textbf{modXHSTT}$.

Instance Creator The Instance Generator isolates the course selection component by modeling only modular events, while remaining timetable structure is represented through resource availability constraints. Instances are defined by 38 parameters describing class structure, course alternatives, student selections, timetable restrictions, and constraint weights.

Generation follows the five steps shown in Figure 2, culminating in the creation of student course selections. Using Latin hypercube sampling over the parameter space, 999 synthetic instances were produced. Parameter ranges were chosen to avoid nonsensical configurations, but feasibility is not enforced, reflect-

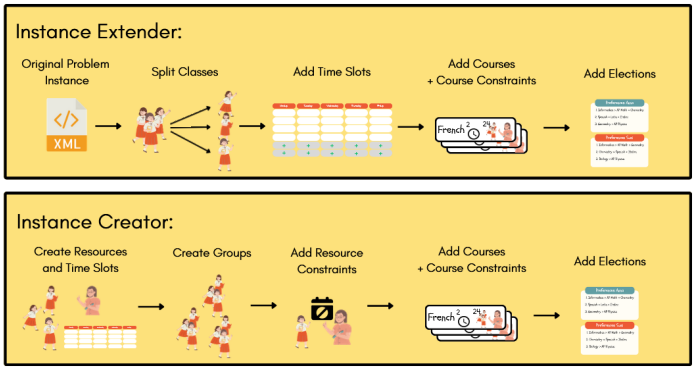


Fig. 2: Instance Generator and Instance Extender pipeline.

ing practical timetabling settings and enabling analysis of feasibility boundaries within the instance space.

Instance Extender The Instance Extender applies a similar procedure to existing benchmark data. We extend 30 real-world XHSTT instances from the Third International Timetabling Competition (ITC 2011) [5] by introducing student-level course selections. In some configurations, we introduce additional timetable capacity (such as extra timeslots) alongside the new events, while in others, the capacity remains unchanged. This allows us to systematically analyze how different solvers handle varying degrees of schedule tightness and complexity, rather than simply easing the baseline optimization problem.

A reduced parameter set (17 parameters) governs course creation and student allocation, while structural characteristics of the base instance are preserved. Classes are subdivided into individual students before courses and selections are generated following the workflow depicted in Figure 2.

For each base instance, 20 parameter configurations were generated—half shared across instances to support comparability and half independently sampled to increase diversity. After removing structurally invalid cases, the final data set comprises 520 extended instances.

4.2 Algorithm Space $\mathcal{A}$

To ensure meaningful differentiation in algorithm behavior, the algorithm space should include methods based on distinct paradigms. We consider two approaches: a SA heuristic [4], specifically designed for modular XHSTT, and an ALNS developed in this work. The ALNS utilizes the exact integer linear programming (ILP) formulation introduced by [3] without modification, embedding this established model within a large neighborhood search that incorporates ideas from LNS approaches for XHSTT [1, 10]. Both algorithms start from the same initial solution generated by the construction heuristic of [4]. Due to the high computational cost, only a single run per instance is performed.

Simulated Annealing The SA approach employs nine neighborhoods that modify times, resources, or sub-event structures. Neighborhoods are selected probabilistically using weights obtained via automatic tuning with SMAC3 [2]. We adapt the original configuration [4] by reducing the number of iterations from 3 million to 1 million to make large-scale experimentation feasible.

Adaptive Large Neighborhood Search The ALNS uses six destroy neighborhoods targeting different aspects of the schedule: (i) removal of times for sub-events without time assignment together with random sub-events, (ii) removal of resources for sub-events without resource assignment together with random sub-events, (iii) removal of resources based on selected timeslots, (iv) removal of sub-events time assignments based on selected resources, (v) removal of times for all sub-events of chosen random events, and (vi) removal of both time and resource assignments for all sub-events of chosen random events.

Reconstruction is performed using Gurobi 13.0 and the exact ILP model of [3]. To execute a repair step efficiently, the intact portions of the schedule are passed to the solver as fixed constants. Specifically, for sub-events whose time assignments are preserved by the destroy neighborhood, the binary start-time variables $y_{se,t}$ are fixed to their current values. Likewise, for preserved resource allocations, the resource assignment variables $w_{se,er,r}$ are locked. This dramatically restricts the active search space, allowing Gurobi to focus exclusively on re-optimizing the unassigned or cleared decision variables.

The destroy size is adapted dynamically based on a target reconstruction time t: it is increased if reconstruction is faster than $(1-\epsilon)t$ and decreased if it exceeds $(1+\epsilon)t$. Neighborhood selection is initially uniform and later guided by a multi-armed bandit strategy with softmax action selection [9], where rewards correspond to objective improvements.

To maintain tractability, neighborhoods do not modify sub-event structures, as this significantly increases ILP complexity. Parameters were tuned using SMAC3 [2], and a runtime limit of 3 hours per instance is imposed.

4.3 Feature Space

To enable ISA, each modular XHSTT instance is mapped to a numerical feature vector describing structural, combinatorial, and resource-related properties of the timetabling problem. The resulting feature space captures both classical instance size characteristics and modularity-specific properties arising from student choice and class formation mechanisms.

We extract a total of 478 features that are grouped into six complementary categories:

- **Instance Scale and Temporal Structure:** Basic structural characteristics describe the overall size of an instance. These include the numbers and proportions of events, students, teachers, classes, rooms, resource types, and available time slots. The temporal structure is further described by the

number of planning days and the mean and variance of time slots per day. Workload-related indicators are derived from the total event duration, the numbers of assigned and unassigned event–resource relations, and the proportion of events with preassigned times.

- **Modularity and Student Choice Structure:** We describe modularity using features such as the proportion of modular events, the number of student positions within modular events, student participation in choice constraints, and structural properties of choice sets. Choice flexibility is measured by statistics on the minimum and maximum required selections, as well as a normalized flexibility ratio $\frac{\text{max choices}-\text{min choices}}{\text{max choices}}$, which captures the degree of freedom available to students. Additional aggregated indicators measure the number of available courses per class group (i.e., groups of classes selecting from the same courses) and per student, as well as the variability of possible course combinations. These features approximate the combinatorial uncertainty present before timetable construction.
- **Class Formation and Balancing:** Modular courses often require dynamic class formation. This aspect is represented by estimates of minimum and maximum feasible enrollments derived from mandatory and optional student assignment roles. Balancing requirements are described by the number of class size balancing constraints, the allowed size deviations, and the proportion of events subject to balancing. Together, these features indicate how strongly enrollment equality restricts feasible allocations.
- **Resource Characteristics and Utilization:** Resource pressure is approximated using ratios between events and available resources (teachers, students, classes, and rooms). Additional indicators include the average number of events per resource, student–teacher and student–room ratios, and the variability of event resource requirements. Expected workload distributions are estimated from preassigned resources, mandatory unavailability, and required assignments, resulting in mean workloads per resource type as well as measures of timetable slack and workload distribution across individual resources. These features approximate congestion and competition for limited resources.
- **Constraint Density and Difficulty Indicators:** Constraint-based characteristics describe how strongly feasibility is restricted. For each constraint type, statistics over hard and soft constraints are extracted, including weight distributions and applied cost functions. Additional indicators measure the proportion of hard constraints, the linkage between events caused by constraints, the concentration of temporal preferences across time slots, and the duration distributions of sub-events under hard feasibility assumptions. These measures serve as indicators of structural difficulty and search space restriction.
- **Graph-Based Structural Features:** To capture higher-order interactions, each instance is represented as a heterogeneous relation graph containing event, resource, time, and constraint nodes. Edges represent assignment relations, temporal adjacency, grouping relations, and constraint applicability. From this representation, the following graph-theoretic descriptors are

extracted: graph size and density, degree statistics per node type, proportions of semantic edge categories, the number and size of connected components, and degree centrality statistics. In addition, an event–resource conflict graph and an event–time conflict graph are analyzed using density, degree distribution, and clustering coefficients to quantify competition between events. Disconnected components indicate decomposable subproblems, while high centrality values reveal structural bottlenecks.

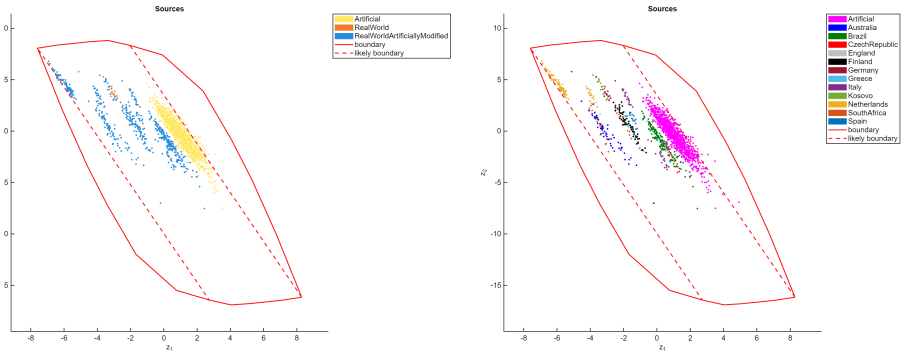
Overall, the resulting feature space jointly captures instance scale, elective modularity, resource contention, constraint tightness, and interaction topology, providing a comprehensive numerical characterization suitable for ISA.

5 Results and Evaluation

In this section, we present the visualized results of the ISA obtained using the MATILDA toolkit[3]. All results were generated with the offline MATLAB implementation, using the recommended parameter settings based on the number of instances. Only minor modifications were made to incorporate the bounds displayed in the online version of the toolkit.

The evaluation is based on a relative comparison of the algorithms, where a performance difference within 5% is considered indicative of comparable (i.e., good) performance.

5.1 Instance distribution

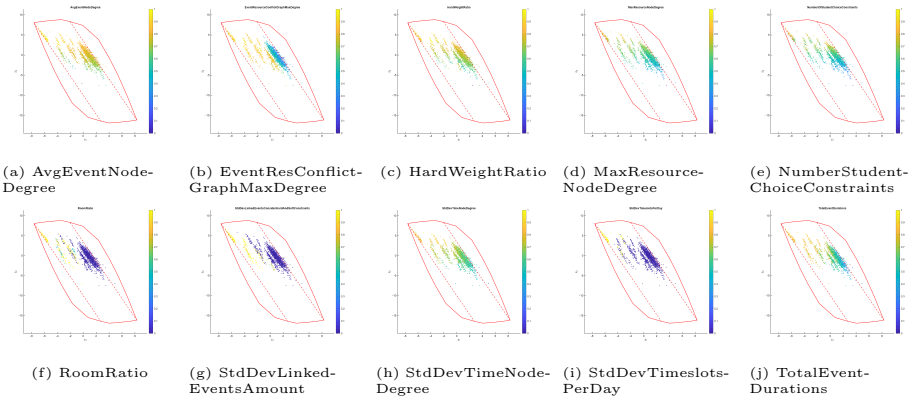


(a) Colored based on method of generation.

(b) Colored based on country of origin.

Fig. 3: Sources used for ISA visualized.

[3] https://matilda.unimelb.edu.au



(a) AvgEventNode-Degree (b) EventResConflict-GraphMaxDegree (c) HardWeightRatio (d) MaxResource-NodeDegree (e) NumberStudent-ChoiceConstraints

(f) RoomRatio (g) StdDevLinked-EventsAmount (h) StdDevTimeNode-Degree (i) StdDevTimeslots-PerDay (j) TotalEvent-Durations

Fig. 4: Features used for generating instance space.

Figure 3 illustrates the distribution of instances in the instance space, revealing several distinct clusters. The artificial instances form a single large cluster, which is expected given their controlled generation process.

The real-world instances from Germany are largely grouped together and are reasonably well covered by modified instances from other countries, indicating that the modification approach can produce instances that resemble real-world data. However, a small number of outliers remain among both real-world and modified instances, suggesting that further diversification could be beneficial.

Artificially modified real-world instances also exhibit a structured pattern: instances derived from the same base instance tend to form local clusters. This suggests that relative solver performance is influenced more strongly by the base instance than by the applied modifications. However, that statement is somewhat relativized considering the strong correlation of the number of student choice constraints with the best performing Algorithm described in Section 5.2.

The projection onto the two-dimensional space shown in Figure 3 and subsequent figures is defined by Equation (1).

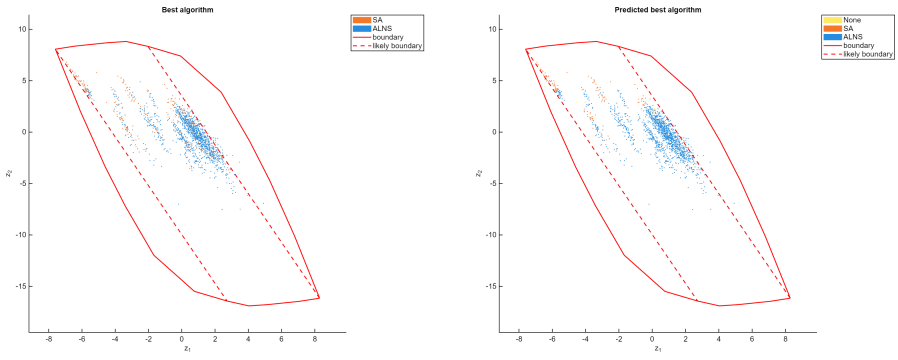
$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} -0.3938 & -0.043 \\ -0.3421 & 0.1236 \\ -0.3029 & 0.1451 \\ -0.2087 & 0.5812 \\ -0.41 & -0.1036 \\ -0.126 & 0.5067 \\ -0.1294 & 0.4098 \\ -0.2404 & 0.3463 \\ -0.1212 & 0.4381 \\ -0.4328 & -0.0724 \end{bmatrix}^T \cdot \begin{bmatrix} RoomRatio \\ StdDevTimeslotsPerDay \\ TotalEventDurations \\ NumberOfStudentChoiceConstraints \\ StdDevLinkedEventsAmount \\ HardWeightRatio \\ MaxResourceNodeDegree \\ StdDevTimeNodeDegree \\ AvgEventNodeDegree \\ EventResConflictGraphMaxDegree \end{bmatrix} \tag{1}$$

Figure 4 provides a more detailed view of the feature distributions. While most features vary across the data sets, some take constant values for specific

subsets. In particular, the RoomRatio (i.e., the ratio of rooms to all resources) is zero for all artificial instances and some modified instances. Similarly, artificial and some modified instances do not include linked events or use equal-length days, resulting in zero variance for these features.

Finally, the EventResConflictGraphMaxDegree is noticeably smaller for artificial instances, due to the absence of explicitly assigned student resources, which reduces the number of resource assignments and thus the degree of event nodes in the conflict graph.

5.2 Algorithm Selection



(a) Best performing algorithm. (b) SVM based algorithm selection.

Fig. 5: Performance of algorithms visualized.

Figure 5 illustrates the performance of the two algorithms across the instance space. The ALNS approach achieves good performance (defined as within 5% of the best-known objective value) on 80.8% of the instances, whereas the SA approach attains this level on 27.6% of instances. By selecting algorithms using the SVM-based approach, the proportion of instances with good performance increases to 86.7%. It is noteworthy that the region in which SA performs well exhibits a substantial overlap with the real-world instances. This makes SA a valuable component of the algorithm portfolio, despite being preferred for a smaller subset of instances overall.

An analysis of the feature space indicates that the number of student choice constraints and the maximum resource node degree are strongly correlated with algorithm performance. In particular, instances with a higher number of student choices and a larger maximum resource node degree tend to favor the SA approach. We attribute this to the increased rigidity of the schedule induced by these features, which appears to negatively impact ALNS. In contrast, SA

benefits from a dedicated mechanism for modifying resources assigned to event groups, which are directly associated with student choice constraints.

6 Conclusion and Future Work

In this paper, we applied ISA to the modular XHSTT problem using a set of 478 newly designed features. We investigated two metaheuristic approaches, namely SA and ALNS, and identified features that are indicative of the relative performance of each method. This analysis highlights the respective strengths and limitations of both approaches. To the best of our knowledge, this work also represents the first application of an ALNS-based method to the modular XHSTT problem.

In addition, we introduced two novel approaches for generating artificial instances of the modular XHSTT problem. The first approach generates instances from scratch, while the second modifies existing XHSTT instances representing real-world schools. This enables a controlled investigation of how the introduction of modular courses affects the structure of instances and the resulting scheduling difficulty.

For future work, several directions appear promising. First, the instance space could be extended by incorporating additional instances as well as a broader range of algorithms. In particular, it would be of interest to study the performance of exact or hybrid methods across the instance space, and to compare their behavior with the metaheuristics considered in this work.

Second, the diversity of real-world instances remains a limiting factor. Expanding the dataset with more real-world modular XHSTT instances would improve the representativeness of the instance space and strengthen the validity of the conclusions. Furthermore, since the XHSTT format natively supports the flexible addition or removal of constraints, future work will evaluate algorithm performance directly on the original, unmodified benchmark instances to establish a clear baseline before modular extensions are applied.

Third, a deeper study into the interplay between algorithmic mechanics and instance characteristics is warranted to explain the performance differences observed in this study. Specifically, future work will investigate the essential differences between real-world and artificial instances that seem to give SA an advantage. This includes a closer examination of why SA's dedicated mechanism for modifying resources assigned to event groups benefits it so heavily when student choices increase and how the ALNS framework might be enhanced with similar neighborhood structures.

Finally, while already extensive, further refinement of the feature set may provide additional insights.

References

1. Demirovic, E., Musliu, N.: Maxsat-based large neighborhood search for high school timetabling. Comput. Oper. Res. **78**, 172–180 (2017). https://doi.org/10.1016/J.COR.2016.08.004, https://doi.org/10.1016/j.cor.2016.08.004

2. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Coello, C.A.C. (ed.) Learning and Intelligent Optimization. pp. 507–523. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
3. Krystallidis, A., Ruiz-Torrubiano, R.: Introducing individuality into students' high school timetables. In: Proceedings of the 14th International Conference on the Practice and Theory of Automated Timetabling-PATAT. vol. 3 (2024)
4. Krystallidis, A., Ruiz-Torrubiano, R., Musliu, N., Dhungana, D.: Integrating course elections systems into high school timetabling through simulated annealing (2026). https://doi.org/10.2139/ssrn.6237970, https://doi.org/10.2139/ssrn.6237970
5. Post, G., Gaspero, L.D., Kingston, J.H., McCollum, B., Schaerf, A.: The third international timetabling competition. Ann. Oper. Res. **239**(1), 69–75 (2016). https://doi.org/10.1007/S10479-013-1340-5, https://doi.org/10.1007/s10479-013-1340-5
6. Post, G., Kingston, J.H., Ahmadi, S., Daskalaki, S., Gogos, C., Kyngäs, J., Nurmi, C., Musliu, N., Pillay, N., Santos, H.G., Schaerf, A.: XHSTT: an XML archive for high school timetabling problems in different countries. Ann. Oper. Res. **218**(1), 295–301 (2014). https://doi.org/10.1007/S10479-011-1012-2, https://doi.org/10.1007/s10479-011-1012-2
7. Rice, J.R.: The algorithm selection problem. Adv. Comput. **15**, 65–118 (1976). https://doi.org/10.1016/S0065-2458(08)60520-3, https://doi.org/10.1016/S0065-2458(08)60520-3
8. Smith-Miles, K., Muñoz, M.A.: Instance space analysis for algorithm testing: Methodology and software tools **55**(12) (2023). https://doi.org/10.1145/3572895, https://doi.org/10.1145/3572895
9. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, second edn. (2018), http://incompleteideas.net/book/the-book-2nd.html
10. Sørensen, M., Kristiansen, S., Stidsen, T.R.: International timetabling competition 2011: An adaptive large neighborhood search algorithm. p. 489 – 492 (2012), https://www.scopus.com/inward/record.uri?eid=2-s2.0-85085570552&partnerID=40&md5=41ddb143588bf6807f37978eb256001c, cited by: 26

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Establishing a National Academic Timetabling Coordination Framework:

The Case of the Forum for Tanzania Academic Timetable Coordinators (FTATC)

Alfred John Nchimbi, Neema Penance Kumburu, and Victoria Theophil Kessy

Forum for Tanzania Academic Timetable Coordinators (FTATC), Tanzania

Abstract

Academic timetabling is one of the most complex and resource-sensitive administrative functions in education. Despite significant advances in automated scheduling and optimization techniques, many developing countries continue to struggle with decentralized practices, limited technical coordination, and uneven capacity among timetable coordinators. This paper presents the establishment and emerging role of the Forum for Tanzania Academic Timetable Coordinators (FTATC) a national technical platform created to strengthen coordination, build capacity, and promote best practices in academic timetabling across all education levels, from pre-primary through university. The initiative is among the first structured national efforts in Sub-Saharan Africa to institutionalize collaboration among timetable coordinators. Using qualitative document analysis and practitioner insights, the paper explores the context that motivated FTATC's creation, its organizational model, capacity-building activities, and its role in professionalizing academic timetabling. Early outcomes include improved technical awareness, stronger inter-institutional coordination, and growing recognition of timetabling as a strategic function. The study contributes to the timetabling literature by presenting a national coordination framework that complements algorithmic and institutional approaches and offers scalable, collaborative lessons for other developing countries.

Keywords: *academic timetabling, FTATC, Tanzania, national coordination, capacity building, professionalisation, Sub-Saharan Africa*

1. Introduction

Academic timetabling is a combinatorial and organizational challenge that requires assigning classes, instructors, students, and rooms to specific time slots while balancing a wide range of constraints including room capacities, instructor availability, course conflicts, and institutional policies. It is considered combinatorial because countless possible arrangements exist, and the goal is to find a schedule that avoids conflicts and makes the best use of available resources.

A growing body of literature has advanced the technical side of this problem. Müller et al. (2025) developed realistic datasets integrating student sectioning with time and room allocation for university course timetabling, demonstrating that combining these elements improves the practical applicability of optimization models. Burke et al. (2007) focused on examination timetabling, showing that distributing exams well across a student's schedule reduces stress and conflicts. Hoshino and Albers (2024) used Integer Linear Programming (ILP) to produce timetables satisfying all hard constraints in under 60 seconds, illustrating the potential of exact optimization methods in structured school settings. Lemos (2019) addressed room allocation specifically, proposing a two-stage ILP model and a greedy heuristic to improve space utilization. Across these studies, a common thread is the focus on algorithmic efficiency yet their real-world applicability depends on institutional conditions that are rarely examined in the literature.

In Tanzania, academic timetabling remains largely decentralized and institution-specific. Most institutions prepare schedules using manual or semi-automated approaches, often handled by individual coordinators who lack formal training, standardized tools, or national guidelines (Maro & Mtebe, 2019; Mtebe & Raisamo, 2014). While technology adoption in education has expanded, persistent barriers including limited technical capacity, weak infrastructure, and uneven use of optimization tools continue to affect scheduling quality (Tanzania Commission for Universities [TCU], 2021; UNESCO, 2020). There is currently no national framework or policy to guide timetabling across educational institutions, and no formal professional pathway for the people who do this work. Institutions operate in isolation, resulting in fragmented practices, duplication of effort, and suboptimal use of resources a pattern consistent with broader findings on how weak administrative coordination undermines educational effectiveness in developing countries (World Bank, 2018).

In response, the Forum for Tanzania Academic Timetable Coordinators (FTATC) was established as a national professional and technical platform to address these gaps. The Forum traces its origins to a training programme conducted under the School of Library, Archives and Documentation Studies (SLADS) during the 2022/2023 academic year. Participants recognized the need for a permanent coordination structure, and four founding members Dr. Alfred John Nchimbi (Open University of Tanzania), Professor Allen Mushi (Mzumbe University), Deusdedith Youuze (Moshi Co-operative University), and Victoria Kessy (SLADS) formally established FTATC. It is now a legally registered non-governmental organization (NGO No. 23929), operating under Tanzania's legal framework and the Ministry of Home Affairs.

FTATC's mandate includes organizing training on best timetabling practices, facilitating related research, providing advisory and consultancy services to educational institutions, collaborating with stakeholders, undertaking advocacy with relevant authorities, and promoting modern technologies in academic scheduling.

Membership has grown from 19 members in 2024 to 40 in 2025 a 110.5% increase and approximately 90 participants have benefited from Forum training programmes to date. This paper examines the Forum's emergence, organizational model, activities, and early outcomes, and situates it within both national and international contexts. It argues that sustainable improvement in academic timetabling requires not only algorithmic sophistication but also the kind of structured national coordination that FTATC is beginning to provide.

2. Literature Review

2.1 Algorithmic and Automated Timetabling Advances

Research on automated timetabling has produced a rich repertoire of computational strategies. Heuristic methods use problem-specific knowledge to generate good solutions quickly, though they risk converging prematurely on suboptimal results (Pearl, 1984). Metaheuristic approaches such as simulated annealing, genetic algorithms, and tabu search balance exploration and exploitation more effectively, often achieving higher solution quality within acceptable timeframes (Blum & Roli, 2003; Di Gaspero & Schaerf, 2000). Hybrid strategies combine the strengths of multiple methods to tackle the most difficult instances (Talbi, 2002; McCollum et al., 2010).

These approaches have been evaluated extensively under assumptions of stable data, well-defined problem structures, and supportive technical environments. In practice, however, noisy or frequently changing data can reduce heuristic reliability (Pearl, 1984). Metaheuristics are often tested in controlled settings by skilled operators using standardized datasets (Blum & Roli, 2003). Hybrid methods, though powerful, assume a level of institutional ICT readiness that may not exist in many developing-country contexts (Talbi, 2002). These assumptions limit how directly advanced optimization findings transfer to settings such as Tanzania's.

2.2 Institutional and Organizational Dimensions of Timetabling

While algorithmic research has flourished, much less attention has been paid to the organizational conditions that enable or constrain effective scheduling. Many institutions lack clear timetabling policies, standard procedures, or accountability frameworks, making decision-making ad hoc and dependent on individual staff rather than institutional systems (World Bank, 2018). Without structured governance, even well-designed algorithms may fail in practice.

Human capacity is another underexplored dimension. Timetabling responsibilities are frequently assigned to academic or administrative staff without formal training in scheduling methodologies or optimization tools (Mtebe & Raisamo, 2014; UNESCO, 2020). Studies on ICT adoption show that capacity is a key factor in successful technology implementation, yet this insight rarely informs timetabling research. A further gap is the assumption of institutional isolation: most timetabling models treat scheduling as a single-institution problem, overlooking the need for cross-institutional

coordination in national education systems. Education system analyses consistently emphasize that coordination mechanisms are essential for improving institutional performance and resource utilization (World Bank, 2018), yet this perspective remains largely absent from the timetabling literature.

2.3 Tanzania's Education System: Structure and Context

Tanzania's education system spans pre-primary, primary, secondary, and post-secondary levels, governed by multiple regulatory bodies and policy frameworks (United Republic of Tanzania [URT], 2023). Pre-primary through secondary education falls under the Ministry of Education, Science and Technology and local government authorities; adult education is coordinated through the Ministry in collaboration with the Institute of Adult Education; vocational training is overseen by the National Council for Technical and Vocational Education and Training (NACTVET); and universities are regulated by the Tanzania Commission for Universities (TCU).

This multi-layered structure creates a complex environment for academic scheduling. Universities generally enjoy greater autonomy in designing academic programmes and timetables, while primary and secondary schools follow more centralized guidelines. As a result, scheduling practices vary considerably, with some institutions adopting flexible technology-supported approaches and others relying on standardized or manual systems (Mtebe & Raisamo, 2014). Rapid enrolment growth in recent years has placed additional pressure on classrooms, teaching staff, and scheduling systems across all levels (UNESCO, 2020; World Bank, 2018), making effective timetabling coordination more urgent than ever.

2.4 Operational Challenges in Tanzanian Timetabling

Despite advances in optimization techniques, much timetabling research pays limited attention to real-world operational environments (Burke et al., 2010; Schaerf, 1999). This gap is especially pronounced in developing countries. In Tanzania, decentralized governance reinforces fragmented scheduling practices (URT, 2023). Most institutions still rely on spreadsheet- or paper-based systems that are time-consuming and error-prone (Mtebe & Raisamo, 2014). A study by Kayanda et al. (2019) found that even where digital tools had been introduced in Tanzanian higher education, accessing and updating timetable data required substantial manual effort, limiting their usefulness for decision-making.

Limited formal training for coordinators compounds these difficulties. Without systematic preparation, staff rely on experience and trial-and-error rather than principled scheduling techniques. Resource constraints further complicate matters: rapid enrolment growth has created classroom shortages, lecturer workload imbalances, and conflicts over shared facilities such as laboratories and specialized rooms (Oude Vrielink et al., 2019; URT, 2025; World Bank, 2018). Finally, timetabling is rarely treated as a strategic function. It tends to be viewed as routine administration rather than an integral part of resource planning—despite evidence that

scheduling decisions have major implications for cost efficiency, teaching quality, and student experience (Lindahl et al., 2018).

2.5 Towards a National Coordination Framework: Conceptual Foundations

This study conceptualizes academic timetabling as a national operational ecosystem in which institutions, regulators, coordinators, and technologies interact within shared constraints, resources, and objectives. Rather than treating scheduling as an isolated institutional task, we view it as a system-wide coordination problem requiring alignment across organizational and governance levels. Drawing on Malone and Crowston's (1994) coordination theory, effective timetabling depends on managing interdependencies among tasks, actors, and resources through clear rules, roles, and communication channels.

The framework is also grounded in professionalisation theory, which holds that occupations evolve into recognized professions through specialized knowledge, formal roles, training systems, and shared standards (Abbott, 1988). FTATC advances this by building capacity, supporting research, and promoting modern technologies progressively transforming timetabling from an ad hoc administrative task into a recognized professional function. A third theoretical anchor is the socio-technical systems perspective, which highlights the interdependence between technological tools and social structures (Baxter & Sommerville, 2011). In Tanzania, where manual and digital systems coexist, successful timetabling innovation requires integrating human capacity, institutional practices, and technological systems simultaneously (Mtebe & Raisamo, 2014; UNESCO, 2020).

2.6 Structural Components of the Framework

The proposed National Academic Timetabling Coordination Framework operates across three interconnected levels. At the institutional level, schools, colleges, and universities prepare and implement timetables, managing data on courses, staff availability, and room capacity. Most existing optimization research is focused at this level but when institutions operate in isolation, fragmentation and inefficiency are the predictable result.

The second level is national coordination, represented by FTATC. The Forum brings together coordinators across institutions, provides training and capacity development, facilitates research and knowledge exchange, promotes modern timetabling technologies, and supports the development of best-practice guidelines. This directly addresses the cross-institution coordination and professionalization gaps identified in the literature.

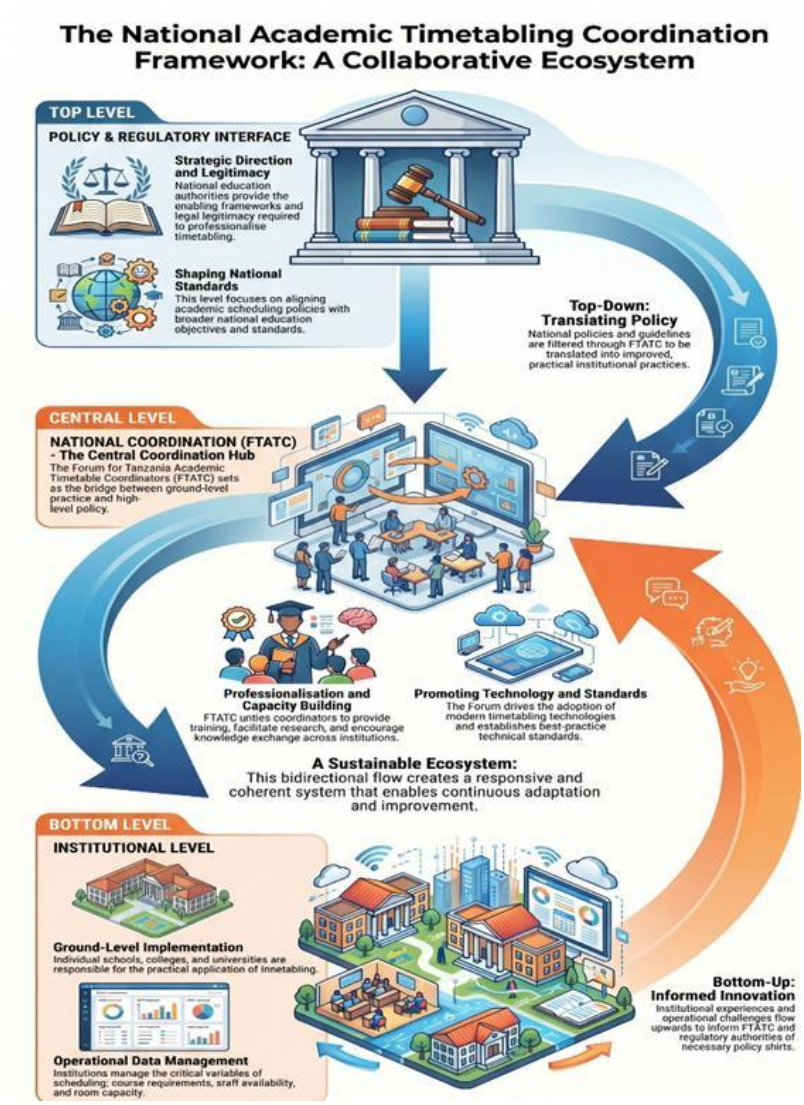


Figure 1: The National Academic Timetabling Coordination Framework — A Collaborative Ecosystem

The third level is the policy and regulatory interface, which links the coordination platform with national education authorities. This level aligns timetabling practices with national education objectives and creates pathways for developing standards and guidelines for academic scheduling. FTATC's constitution explicitly mandates advocacy and collaboration with education stakeholders, positioning the Forum as a bridge between practice and policy.

The framework's strength lies in its bidirectional integration. A bottom-up flow allows institutional experiences and practical innovations to inform national coordination. A top-down flow ensures that policies and standards developed at the regulatory level are translated through FTATC into improved institutional practices. FTATC functions at the centre as a coordination hub, synthesizing experiences, building capacity, and aligning policy and practice creating a feedback-driven socio-technical ecosystem that is both responsive and sustainable.

3. Methodology

3.1 Research Design

This study employed a qualitative document-based case study design to examine the establishment, development, and strategic potential of FTATC as a national platform for strengthening academic timetabling in Tanzania. The qualitative approach was appropriate because the study sought to understand institutional processes, organizational development, stakeholder perspectives, and governance arrangements rather than to measure numerical outcomes (Creswell & Poth, 2016; Yin, 2018). An institutional development lens guided the exploration of how FTATC emerged, evolved, and positioned itself as a coordinating mechanism across higher education institutions.

3.2 Data Sources

The study relied exclusively on documentary evidence generated during the establishment and operation of FTATC. Documents were purposively selected based on their relevance to the Forum's formation, governance, activities, stakeholder engagement, and strategic direction. As Bowen (2009) notes, document analysis provides a clear and concise picture of existing evidence and is especially valuable when empirical data from participants is unavailable or supplementary. Seven categories of documents were used, as summarized in Table 1.

Table 1: Documentary Sources Used in the Study

Data Source	No. of Documents	Period Covered	Purpose
Founding documents and concept notes	2	2024–2025	Understanding Forum establishment
Constitution and governance documents	2	2024–2025	Examining organizational structure
Meeting minutes	20	2024–2025	Tracing institutional development
Membership records	1	2024–2025	Assessing growth and participation
Official correspondence	20	2024–2025	Examining stakeholder engagement

Data Source	No. of Documents	Period Covered	Purpose
Strategic plans and reports	22	2024–2025	Identifying priorities and future directions
Conference presentations and communication materials	5	2024–2025	Understanding advocacy and outreach efforts

3.3 Data Analysis

This study integrated three complementary analytical strategies: thematic analysis, institutional process tracing, and framework development synthesis.

3.3.1 Thematic Analysis

Thematic analysis was used to identify recurring patterns, issues, and priorities in the FTATC documentary evidence, following the six-stage process described by Naeem et al. (2023) and adapted in line with approaches used in timetabling research (Xu & Zammit, 2020): (1) familiarization with the documents; (2) initial coding of relevant text segments; (3) identification of recurring patterns; (4) development of preliminary themes; (5) review and refinement of themes; and (6) interpretation and reporting. Key themes that emerged included the professionalization of academic timetabling, institutional coordination, capacity development, governance mechanisms, and international collaboration. Table 2 illustrates the coding and thematic development process.

Table 2: Sample of Thematic Analysis Coding

Information Extracted	Initial Code	Theme
“There is a need to bring timetable coordinators together nationally.”	National networking	Coordination and Collaboration
“Training opportunities are required for timetable officers.”	Skills development	Capacity Building
“Institutions currently operate independently without common standards.”	Lack of harmonization	Governance and Standardization
“The Forum should engage international partners.”	External collaboration	International Engagement

3.3.2 Institutional Process Tracing

Process tracing was used to reconstruct the chronological development of FTATC and identify critical events that shaped its evolution (Gatti, 2025; George & Bennett, 2005). Evidence from meeting minutes, correspondence, governance documents, and strategic

records was systematically analyzed to establish developmental linkages, focusing on founding events, organizational milestones, membership growth, governance developments, and national and international engagement activities. Table 3 presents the key chronological events.

Table 3: Institutional Process Tracing Timeline

Period	Key Event	Evidence Source	Significance
2022/2023	Training programme under SLADS initiates coordination discussions	Founding documents	Genesis of the initiative
2024	Formal establishment and NGO registration of FTATC	Founding documents	Official institutionalization
2024	Initial membership of 19 coordinators from multiple institutions	Membership records	National reach established
2025	Membership grows to 40; approximately 90 trained participants	Membership records	Expanding national impact
2024–2025	International engagement activities initiated	Correspondence and reports	Global visibility

3.3.3 Framework Development Synthesis

Framework development synthesis was used to integrate empirical findings into a proposed national coordination framework (De Padua et al., 2026). Drawing on insights from thematic analysis and process tracing, the approach enabled the systematic consolidation of institutional challenges into coherent strategic priorities and organizational components. Table 4 maps empirical findings to framework elements.

Table 4: Framework Development Synthesis

Empirical Finding	Strategic Need	Proposed Framework Component
Limited professional training	Capacity enhancement	Capacity Building Unit
Weak inter-institutional coordination	National collaboration	Coordination Secretariat
Lack of common standards	Quality assurance	Standards and Guidelines Committee

Empirical Finding	Strategic Need	Proposed Framework Component
Limited international engagement	Global networking	International Relations Desk

3.4 Trustworthiness

Several measures were taken to ensure the credibility and dependability of the study (Islam, 2020; Zulkifli et al., 2024). Data triangulation was achieved through the use of multiple document types—governance records, meeting minutes, official correspondence, and strategic reports—enabling cross-verification of evidence. An audit trail was maintained throughout data collection, coding, and analysis to support transparency. Continuous comparison of interpretations across sources was undertaken to enhance consistency and minimize researcher bias.

3.5 Ethical Considerations

The study used organizational documents obtained through authorized access. Confidential information was excluded from the analysis, and all documentary evidence was used solely for academic purposes. Care was taken to represent organizational events and stakeholder contributions accurately, in accordance with professional and ethical research standards.

4. Findings and Discussion

4.1 Establishment and Organizational Structure of FTATC

The Forum for Tanzania Academic Timetable Coordinators (FTATC) is a legally registered non-governmental organization (NGO No. 23929) operating under Tanzania's NGO Act (URT, 2002) and overseen by the Ministry of Home Affairs. As a nationally recognized association, it brings together academic timetable coordinators from schools, colleges, and universities, functioning as an independent professional body while complying with national legal requirements.

FTATC's mandate positions it as a knowledge-sharing and capacity-building platform that bridges theory and practice. Its key functions include promoting collaboration and knowledge exchange, providing training on best timetabling practices, facilitating research, offering advisory and consultancy services, undertaking advocacy with relevant authorities, and promoting the adoption of modern technologies in academic scheduling (FTATC, 2024). Membership is voluntary and open to individual practitioners directly involved in timetabling, as well as other interested stakeholders—an approach that grounds the Forum in operational realities and ensures it responds to genuine practitioner needs.

4.2 Capacity-Building Activities

Capacity-building workshops are the Forum's primary intervention for strengthening the technical and professional competencies of timetable coordinators. These events

cover foundational and advanced concepts in scheduling, including constraint identification, resource allocation, conflict management, and optimization strategies all of which are essential for producing feasible and efficient timetables in complex educational environments (Schaerf, 1999; Burke et al., 2010).

Workshops also expose participants to automated timetabling systems and software tools, which is particularly relevant in Tanzania, where many institutions still rely on manual or semi-automated scheduling (Mtebe & Raisamo, 2014). Beyond technical training, peer case discussions allow coordinators to share institutional challenges and context-specific solutions, promoting the kind of collective problem-solving that coordination theory identifies as essential for managing interdependent tasks (Malone & Crowston, 1994). Participants are also guided on developing formal timetabling policies and accountability mechanisms within their institutions—a shift from operational scheduling toward structured governance.

Annual General Meetings (AGMs) complement workshop activities, serving as FTATC's supreme decision-making forum and as a space for policy dialogue. Through presentations, case studies, and stakeholder discussions, AGMs facilitate the dissemination of good practices and enable the Forum to contribute to shaping national guidelines on academic timetabling (World Bank, 2018). FTATC's strategy of rotating AGMs and workshops across different institutions has also enhanced its visibility and demonstrated relevance to institutional leaders at multiple levels.

4.3 Professionalizing Academic Timetabling

One of FTATC's most significant strategic interventions has been its work to establish academic timetable coordination as a recognized professional function, rather than an administrative afterthought. The FTATC constitution supports this by bringing together practitioners, providing training, supporting research, and advocating improved timetabling practices across educational institutions. In doing so, the Forum is helping to define the roles, competencies, and professional standards associated with the 'Academic Timetable Coordinator' role.

Professionalisation theory holds that occupations gain professional status through specialized knowledge, formal role recognition, and institutional support (Abbott, 1988). FTATC advances this process by fostering a community of practice, promoting shared standards, and advocating for the institutionalization of the role within Tanzania's education system. The complexity of modern timetabling managing constraints, optimizing resources, and integrating digital tools—demands exactly this kind of formalized expertise.

4.4 Early Outcomes: Technical Awareness and Institutional Coordination

FTATC's activities have produced several notable early outcomes. First, there has been a measurable improvement in technical awareness among coordinators. Through workshops and AGMs since 2022, the Forum has introduced participants to both

foundational and advanced concepts in academic timetabling. Many coordinators who previously relied on intuitive or manual approaches have developed a more structured understanding of constraint modelling—distinguishing hard constraints (such as room capacity and lecturer availability) from soft constraints (such as preferences and workload balance)—and of methods for managing them (Schaerf, 1999; Burke et al., 2010). Exposure to software-based and algorithm-driven scheduling has also broadened participants' awareness of tools available to improve efficiency and reduce conflicts, aligning with broader findings that ICT competence is critical for administrative effectiveness in education systems (UNESCO, 2020).

Second, the Forum has strengthened coordination among timetable practitioners across different levels of the education system. Before FTATC's establishment, many coordinators worked in relative isolation, with limited opportunities for knowledge sharing or professional support. The Forum has reduced this isolation and fostered a sense of community and mutual support. It has also promoted informal benchmarking, exposing coordinators to different approaches for handling constraints, resource allocation, and scheduling conflicts. This kind of peer learning is recognized as a valuable mechanism for continuous improvement and organizational learning (Burke et al., 2010). Communication platforms established through the Forum now allow coordinators to seek technical advice and resolve challenges more rapidly than was previously possible.

Table 5: Summary of Early Outcomes and Observed Changes

Indicator	Before FTATC (2022)	After FTATC (2024–2025)
Institutions represented	12	45
Number of trained coordinators	30	180
Institutions with formal timetable committees	4	15
Average timetable revision rounds per semester	4–6	1–2
Use of software-assisted scheduling	Limited	Increasing adoption
Networking platform for coordinators	None	Established

4.5 Emerging Institutional and Policy Changes

Although FTATC is still relatively young, there are early signs that its activities are beginning to influence institutional practices and policy discussions. A notable shift has been the growing recognition of the need for formal timetabling procedures and dedicated coordination structures within institutions. Some higher learning institutions have begun reviewing their timetable preparation procedures and assigning dedicated

officers or committees to coordinate academic schedules, rather than treating the function as an incidental administrative responsibility.

Timetable management has also begun to feature more prominently in senior management discussions, including meetings of Deputy Vice Chancellors responsible for Academic, Research, and Consultancy (DVC-ARCs). Conversations have focused on improving teaching efficiency, classroom utilization, and minimizing scheduling conflicts. While no national timetabling policy has yet been adopted, these developments indicate an emerging shift toward institutionalizing timetabling as a specialized management function consistent with professionalisation theory's account of how occupational groups gradually gain legitimacy through the establishment of standards and organizational structures (Abbott, 1988).

4.6 Strategic Recognition and Stakeholder Engagement

A significant early outcome of FTATC has been its growing strategic visibility at senior management and policy levels. Since its formal registration in 2024, the Forum has engaged high-level stakeholders, including informal consultations with the then Prime Minister Kassim Majaliwa and the Minister of Education Professor Adolf Mkenda, as well as discussions with TAMISEMI (Tanzania's regional and local government authority), the Ministry of Education, DVC-ARC meetings in public universities, and the Njombe Regional Secretariat.

These engagements have helped position timetabling within broader conversations about resource utilization, teaching effectiveness, and student experience an important reframing consistent with strategic perspectives on educational administration (World Bank, 2018). FTATC has demonstrated its value as a credible link between practitioners and policymakers, facilitating dialogue that is beginning to shift how institutions think about scheduling. The Forum's strategy of rotating AGMs and workshops across different host institutions has further amplified this effect: Moshi Co-operative University and the Njombe Regional Secretariat both provided venues and logistical support free of charge, reflecting genuine institutional confidence in FTATC's contribution. Participation by senior university managers and regional authorities in Forum events has raised the profile of academic timetabling and strengthened its position within broader discussions on academic quality and governance.

Preliminary evidence suggests improvements in communication among coordinators, reductions in repeated timetable adjustments in some institutions, and increased awareness of the value of formal timetabling policies. While systematic impact data is still being generated, these early indicators provide a meaningful foundation for future longitudinal studies.

4.7 Implications for Automated Timetabling

The findings of this study suggest that national coordination frameworks like FTATC play a critical complementary role to algorithmic timetabling approaches, particularly in resource-constrained environments. Effective automated timetabling requires clean, well-structured data yet in many Tanzanian institutions, data management remains fragmented and inconsistent (Mtebe & Raisamo, 2014). It also requires trained operators who understand both the technical and practical dimensions of scheduling, but many coordinators lack this preparation. Furthermore, algorithmic approaches assume stable institutional environments with clear governance structures, which may not exist in highly decentralized systems.

FTATC addresses these preconditions by promoting standardized data practices, building practitioner competencies through training, and fostering the shared standards and policy alignment that automation depends on. In this sense, the Framework's value lies in improving system readiness for automation reinforcing the argument that technical optimization must be supported by strong institutional and coordination mechanisms to achieve meaningful impact in real educational settings.

4.8 Scalability to Other Developing Countries

The National Academic Timetabling Coordination Framework is not only relevant to Tanzania but has potential for adaptation across Sub-Saharan Africa and other developing regions. Its design reflects a low-cost, practitioner-driven, and contextually adaptable model suited to education systems characterized by resource constraints, fragmented coordination, and limited technical capacity.

The framework is low-cost because it relies primarily on existing institutional networks and locally organized activities such as workshops and AGMs, rather than large investments in software or infrastructure—important in contexts where financial and technological resources are limited (World Bank, 2018). It is practitioner-driven, emerging from the collective efforts of coordinators themselves rather than top-down policy directives. This bottom-up character aligns with coordination theory's emphasis on decentralized but structured collaboration (Malone & Crowston, 1994) and supports both sustainability and local ownership. Its core components capacity development, knowledge exchange, standardization, and policy engagement are flexible enough to be tailored to different national environments, while its emphasis on socio-technical integration ensures that both human and technological factors are addressed in implementation (Baxter & Sommerville, 2011).

5. Theoretical Contribution

This study makes a distinct contribution to the Practice and Theory of Automated Timetabling (PATAT) domain by extending timetabling research beyond algorithmic optimization to encompass institutional, organizational, and policy dimensions. While constraint programming, integer linear programming, and metaheuristics have advanced computational efficiency and solution quality within individual institutions (Burke et al., 2010; Schaerf, 1999), this paper demonstrates that their effectiveness

depends on broader systemic conditions including governance structures, human capacity, and coordination mechanisms.

By positioning academic timetabling as a socio-technical and organizational system rather than merely a computational problem, the study highlights the importance of aligning algorithms with institutional practices, user competencies, and policy environments (Baxter & Sommerville, 2011; Malone & Crowston, 1994). It also introduces a national-level coordination dimension that is largely absent from existing timetabling literature. Most PATAT studies assume institution-specific problem settings, with limited attention to how scheduling practices can be harmonized across institutions within a national education system. The FTATC case provides a concrete model for understanding how cross-institutional collaboration and standardization can enhance efficiency and consistency—particularly in developing-country contexts where fragmentation and weak coordination are common. Overall, this shifts the contribution of timetabling research from purely algorithmic optimization toward a combined socio-technical systems perspective.

6. Conclusions

This study reaffirms a central finding: sustainable improvement in academic timetabling requires not only algorithmic innovation but also strong national coordination structures. Although the PATAT research community has made significant advances in optimization techniques and computational models, their effectiveness ultimately depends on supportive institutional, professional, and policy environments (Burke et al., 2010; Schaerf, 1999). Without coordinated systems, trained practitioners, and standardized practices, even sophisticated algorithms may struggle to deliver results in real educational contexts.

The Forum for Tanzania Academic Timetable Coordinators (FTATC) represents an emerging and context-sensitive model of national coordination. Through capacity-building workshops, knowledge-sharing platforms, advocacy, and stakeholder engagement, it demonstrates how practitioner-driven initiatives can address fragmentation, enhance professionalization, and strengthen coordination across institutions. Much of the existing timetabling literature is grounded in contexts with advanced technological infrastructure and stable institutional systems, leaving the challenges faced by developing countries resource constraints, decentralized practices, limited technical capacity—underexplored. This study contributes by presenting a context-specific coordination model grounded in real institutional practice rather than algorithmic development.

Future research should build on these early findings by tracking long-term outcomes through longitudinal studies and quantitative impact assessments. Expanding the geographical diversity of timetabling research to include more developing-country contexts will enrich the field with new perspectives, challenges, and solution pathways.

Finally, greater international collaboration between practitioners and researchers across regions and disciplines is needed to move academic timetabling toward more inclusive, scalable, and practically relevant solutions.

References

Abbott, A. (1988). The system of professions: An essay on the division of expert labor. University of Chicago Press.

Baxter, G., & Sommerville, I. (2011). Socio-technical systems: From design methods to systems engineering. Interacting with Computers, 23(1), 4–17. https://doi.org/10.1016/j.intcom.2010.07.003

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys, 35(3), 268–308. https://doi.org/10.1145/937503.937505

Bowen, G. A. (2009). Document analysis as a qualitative research method. Qualitative Research Journal, 9(2), 27–40. https://doi.org/10.3316/QRJ0902027

Burke, E. K., Mareček, J., Parkes, A. J., & Rudová, H. (2010). Decomposition, reformulation, and diving in university course timetabling. Computers & Operations Research, 37(3), 582–597. https://doi.org/10.1016/j.cor.2009.02.023

Burke, E. K., McCollum, B., Meisels, A., Petrovic, S., & Qu, R. (2007). A graph-based hyper-heuristic for educational timetabling problems. European Journal of Operational Research, 176(1), 177–192. https://doi.org/10.1016/j.ejor.2005.08.012

Creswell, J. W., & Poth, C. N. (2016). Qualitative inquiry and research design: Choosing among five approaches. SAGE Publications.

De Padua, S. I. D., Riffel, M. F., Santos, É. B. E., De Castro, L. S. P., & De Araujo, C. (2026). Integrating business process management with the sustainable development goals: A multidimensional framework for systemic sustainability. Sustainable Development. https://doi.org/10.1002/sd.71082

Di Gaspero, L., & Schaerf, A. (2000). Tabu search techniques for examination timetabling. In E. Burke & W. Erben (Eds.), Practice and theory of automated timetabling III (Lecture Notes in Computer Science, Vol. 2079, pp. 104–117). Springer. https://doi.org/10.1007/3-540-44629-X_7

Forum for Tanzania Academic Timetable Coordinators (FTATC). (2024). The constitution of the Forum for Tanzania Academic Timetable Coordinators (FTATC). Dar es Salaam, Tanzania.

Gatti, F. (2025). The challenging construction of a focal organization: Lessons from the postwar trade complex (1947–1971). Review of International Political Economy, 33, 919–949. https://doi.org/10.1080/09692290.2025.2599254

George, A. L., & Bennett, A. (2005). Case studies and theory development in the social sciences. MIT Press.

Hoshino, R., & Albers, J. (2024). Cohort-based timetabling with integer linear programming. In Proceedings of the 14th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2024).

Islam, M. R. (2020). Reliability and validity in qualitative research. International Journal of Research and Innovation in Social Science, 4(10), 439–443.

Kayanda, A., Busagala, L. S. P., & Tedre, M. (2019). Design and implementation of timetabling software for improved decision making in Tanzanian higher education. IEEE AFRICON 2019 Proceedings. https://doi.org/10.1109/AFRICON46755.2019.9133914

Lemos, A. (2019). Room usage optimization in timetabling: A case study. Computers & Operations Research, 103, 1–12.

Lindahl, M., Mason, A. J., Stidsen, T., & Sørensen, M. (2018). A strategic view of university timetabling. European Journal of Operational Research, 266(1), 35–45. https://doi.org/10.1016/j.ejor.2017.09.022

Malone, T. W., & Crowston, K. (1994). The interdisciplinary study of coordination. ACM Computing Surveys, 26(1), 87–119. https://doi.org/10.1145/174666.174668

Maro, G., & Mtebe, J. (2019). ICT integration in Tanzanian higher education: Trends and challenges. Journal of Information Technology Education, 18, 255–274.

McCollum, B., McMullan, P., Parkes, A. J., Burke, E. K., & Qu, R. (2010). An extended great deluge approach to the examination timetabling problem. European Journal of Operational Research, 204(1), 46–56. https://doi.org/10.1016/j.ejor.2009.10.005

Mtebe, J. S., & Raisamo, R. (2014). Challenges and instructors' intention to adopt and use open educational resources in higher education in Tanzania. The International Review of Research in Open and Distributed Learning, 15(1), 249–271. https://doi.org/10.19173/irrodl.v15i1.1687

Müller, T., Rudová, H., & Müllerová, Z. (2025). Real-world university course timetabling at the International Timetabling Competition 2019. Journal of Scheduling, 28(2), 247–267. https://doi.org/10.1007/s10951-023-00801-w

Naeem, M., Ozuem, W., Howell, K. E., & Ranfagni, S. (2023). A step-by-step process of thematic analysis to develop a conceptual model in qualitative research. International Journal of Qualitative Methods, 22. https://doi.org/10.1177/16094069231205789

Oude Vrielink, R. A., Jansen, E. A., Hans, E. W., & Van Hillegersberg, J. (2019). Practices in timetabling in higher education institutions: A systematic review.

Annals of Operations Research, 275, 145–160. https://doi.org/10.1007/s10479-017-2688-8

Pearl, J. (1984). Heuristics: Intelligent search strategies for computer problem solving. Addison-Wesley.

Schaerf, A. (1999). A survey of automated timetabling. Artificial Intelligence Review, 13(2), 87–127. https://doi.org/10.1023/A:1006576209967

Talbi, E.-G. (2002). A taxonomy of hybrid metaheuristics. Journal of Heuristics, 8(5), 541–564. https://doi.org/10.1023/A:1016540724870

Tanzania Commission for Universities. (2021). University education statistics report. Dar es Salaam: TCU.

UNESCO. (2020). ICT in education in Africa: A comparative analysis of basic e-readiness in schools. Paris: UNESCO.

United Republic of Tanzania. (2002). Non-governmental organizations act, 2002. Dar es Salaam: Government Printer.

United Republic of Tanzania. (2023). Education and training policy (2023 ed.). Ministry of Education, Science and Technology.

United Republic of Tanzania. (2025). Education sector development plan 2025/26–2029/30. Ministry of Education, Science and Technology.

World Bank. (2018). World development report 2018: Learning to realize education's promise. World Bank. https://doi.org/10.1596/978-1-4648-1096-1

Xu, W., & Zammit, K. (2020). Applying thematic analysis to education: A hybrid approach to interpreting data in practitioner research. International Journal of Qualitative Methods, 19. https://doi.org/10.1177/1609406920918810

Yin, R. K. (2018). Case study research and applications: Design and methods (6th ed.). SAGE Publications.

Zulkifli, H., Ismail, A. M., Muhamad, N. A. F., Hussin, N. H., & Mahdi Yusuf, S. A. (2024). Reliability and validity of the fuzzy delphi instrument of Islamic cognitive domains. International Journal of Academic Research in Progressive Education and Development, 13(2), 17551–17578. https://doi.org/10.6007/IJARPED/v13-i2/21201

Weighted Spearman Phenotypic Characterization for Surrogate-Assisted Genetic Programming in Dynamic Vehicle Scheduling

Leshan Tan[1 [0009−0002−6060−3834]], Chenwei Jin[1 [0009−0003−5694−9709]], Xinan Chen[1 [0000−0001−9620−3264]], Rong Qu[2 [0000−0001−8318−7509]], and Ruibin Bai[1⋆ [0000−0003−1722−568X]]

[1] University of Nottingham Ningbo China, Ningbo, China
{leshan.tan, chenwei.jin, xinan.chen, ruibin.bai}@nottingham.edu.cn
[2] University of Nottingham, Nottingham, United Kingdom
rong.qu@nottingham.ac.uk

Abstract. Surrogate-assisted genetic programming (SAGP) has become an effective approach for reducing the computational burden of automated heuristic design in dynamic scheduling, where fitness evaluation is typically dominated by repeated stochastic simulation. Its effectiveness, however, depends critically on the characterization used to represent GP individuals for surrogate modeling. Conventional top-1 phenotypic characterization (PC) retains only a top-1 signal from each decision situation, thereby discarding most of the induced preference ordering and yielding a coarse behavioral representation for distance-based surrogates. This paper proposes weighted Spearman phenotypic characterization (WSPC), which maps each individual to a fixed-length behavioral embedding by comparing the complete ranking it induces in each decision situation with that of a reference policy. WSPC introduces position-dependent weights into the Spearman correlation so that disagreements near the top of the ranking receive greater emphasis. The resulting characterization preserves richer behavioral information than conventional top-1 PC while remaining compatible with standard distance-based surrogates such as k-nearest neighbors. Experiments on dynamic container terminal truck scheduling show that, under the same training-time budget, WSPC attains the best average test performance, with significant gains over standard GP and top-1 PC. Diagnostic surrogate-quality results further support the effectiveness of WSPC for surrogate guidance.

Keywords: Genetic Programming · Surrogate Model · Phenotypic Characterization · Dynamic Scheduling

1 Introduction

Genetic Programming (GP) [13] has demonstrated strong capability in evolving effective heuristics for dynamic scheduling [4, 9]. However, its practical scalability

⋆ Corresponding author.

is fundamentally constrained in simulation-based environments, where fitness evaluation often requires repeated stochastic simulation over many instances. Since a GP run may evaluate thousands of candidate rules, the cumulative computational cost can quickly become prohibitive. To alleviate this bottleneck, surrogate-assisted GP (SAGP) incorporates predictive models to approximate expensive fitness evaluations at a substantially lower cost [7, 19, 24].

The effectiveness of SAGP depends critically on how GP individuals are represented. Most surrogate models used in this setting, including k-nearest neighbors, support vector regression, and Gaussian processes, assume inputs in the form of fixed-length numerical vectors [7, 11]. GP individuals, by contrast, are variable-sized symbolic programs whose syntactic structures and operational semantics cannot be directly processed by standard regression models. Consequently, surrogate quality is determined not only by the predictive model itself but also by the characterization used to represent GP individuals. If the resulting representation is too coarse or unstable, surrogate estimates may become less reliable, distort selection pressure, and ultimately impair search performance.

Existing characterization methods can be broadly categorized into genotypic and phenotypic approaches. Genotypic characterization (GC) extracts features directly from the representation of an individual, such as node frequencies or structural distances [14, 16, 17]. Phenotypic characterization, in contrast, captures the observable decision behavior induced by a rule on sampled decision situations (DSs) [7]. A commonly used phenotypic design in scheduling-oriented SAGP is a top-1 form of characterization (PC), which retains only a top-1 signal from each DS [7, 23]. This design is compact and efficient, but it discards most of the preference structure induced by the rule. As a consequence, two heuristics that agree on the same top-1 signal but differ substantially in the remaining ranking may become indistinguishable. Moreover, the resulting characterization is weakly comparable across DSs with different candidate-set sizes, since the same behavioral discrepancy can be expressed at different scales.

To address these limitations, this paper proposes weighted Spearman phenotypic characterization (WSPC) for SAGP in dynamic scheduling. The central idea is to represent a GP individual by the complete ranking it induces over the candidate actions in each DS and to compare this ranking against that of a fixed reference policy. The resulting per-DS agreement values are then stacked to form a fixed-length behavioral embedding. To reflect the practical importance of early decisions in online scheduling, WSPC incorporates position-dependent weights so that disagreements near the top of the ranking contribute more strongly than those near the tail. In this way, WSPC preserves richer behavioral information than PC while remaining compatible with existing distance-based surrogates.

The major contributions of this paper are as follows:

- We propose weighted Spearman phenotypic characterization (WSPC), a rank-based characterization that encodes the complete preference ordering induced by a GP rule in each decision situation and maps it to a fixed-length behavioral embedding relative to a reference policy.

– We introduce a position-aware weighting scheme that assigns greater importance to top-ranked disagreements, making the representation more sensitive to differences that are more consequential in dynamic scheduling.
– Results on dynamic container terminal truck scheduling show that, under the same training-time budget, WSPC achieves the best average scheduling performance among SGP-PS variants, and surrogate-quality diagnostics show better identification of promising individuals than top-1 PC.

2 Background and Related Work

2.1 Genetic Programming and Surrogate Modeling

Genetic Programming (GP) [13], a widely used hyper-heuristic approach [2, 22], is an evolutionary search paradigm that evolves computer programs or decision rules in a flexible symbolic representation space. In dynamic scheduling, GP individuals are typically interpreted as priority functions: given the current state and feasible actions, the rule assigns a score to each candidate and the scheduler executes the highest-priority action. Due to their interpretability and flexibility, GP-based rules have been applied successfully to a range of dynamic scheduling problems, including job shop scheduling [7, 15, 23], yard crane scheduling [9], and container terminal truck scheduling [3–5]. Related dynamic pickup-and-delivery problems have also motivated learning-based optimization approaches [21].

Despite these advantages, the practical scalability of GP in dynamic scheduling remains constrained by the cost of fitness evaluation. In simulation-based settings, each candidate rule may need to be assessed on multiple stochastic instances to obtain a reliable performance estimate. As population sizes, generation counts, and problem complexity increase, the cumulative computational burden can become prohibitive. This challenge has motivated the incorporation of surrogate models into GP. Surrogate models, also referred to as approximation models or meta-models, are computationally inexpensive predictors used to approximate expensive objective functions [6, 11, 12]. Common surrogate models include k-nearest neighbors, support vector regression, Gaussian processes, radial basis function models, and other similarity-based predictors [11, 12]. Despite their methodological differences, these models share a common assumption: the input is provided in the form of fixed-length numerical vectors [7].

This requirement makes characterization essential in SAGP because GP individuals are variable-sized symbolic structures that are not directly compatible with standard surrogate models. Before surrogate modeling can be performed, each individual must be mapped to an informative fixed-dimensional representation. The practical success of SAGP depends jointly on the predictive model and characterization. If the adopted characterization is coarse, unstable, or weakly aligned with task-specific behavior, proximity in the induced representation may fail to reflect meaningful behavioral or performance similarity, causing surrogate estimates to distort selection pressure and degrade search performance. These issues have motivated extensive research on characterization design.

2.2 Genotypic and Phenotypic Characterization

Characterization methods used in SAGP can be broadly categorized into genotypic and phenotypic approaches. Genotypic characterization (GC) encodes individuals through syntactic or structural properties of the GP representation, such as node frequencies, subtree statistics, or structural distances [14, 16, 17]. Since GC is derived directly from the program tree, it is generally stable across evaluations and can often be computed efficiently. However, for decision-making problems such as dynamic scheduling, structural similarity is not necessarily behavior-preserving: syntactically distinct rules may induce similar decisions, whereas small structural differences may produce markedly different scheduling trajectories. As a result, GC-based distances may correlate only weakly with true fitness differences, making them a less reliable basis for surrogate modeling.

Phenotypic characterization, in contrast, represents individuals through their observable behavior on decision situations [7, 20]. In scheduling applications, this behavior is reflected in how a rule scores, ranks, or selects candidate tasks in representative states. Such representations enable the surrogate to model similarity in terms of induced decisions rather than program syntax. In scheduling-oriented SAGP, a commonly used phenotypic form is the top-1 characterization (PC), which retains only a top-1 signal from each DS [7, 23]. Recent work has also combined GC and PC to enhance surrogate guidance [19, 24]. Nevertheless, PC remains the component encoding decision behavior in such hybrid designs, so a top-1 phenotypic component still carries substantial information loss.

2.3 Dynamic Container Terminal Truck Scheduling

We adopt dynamic container terminal truck scheduling (DCTTS) as the primary testbed in this study. A typical container terminal comprises the berth area, where quay cranes transfer containers between vessels and trucks, the yard area, where yard cranes manage storage and retrieval operations within yard blocks, and the entry-exit area, which connects the terminal to the external transportation network. Internal trucks move containers between berth-side and yard-side resources, thereby coupling these operational areas. DCTTS concerns the real-time assignment of internal trucks to transport tasks under dynamic task availability, stochastic service times, and uncertain travel conditions [3, 4, 10]. Since trucks coordinate the interactions among the terminal's principal resources, effective truck dispatching is critical for maintaining high resource utilization and terminal throughput, typically measured in TEU/h.

The DCTTS environment is simulated through an event-driven simulator, as illustrated in Fig. 1. During simulation, a decision situation (DS) is triggered whenever a truck becomes idle. In each DS, the dispatching rule determines the truck's next task from the feasible candidate set defined by the current system state. Let $\Omega(d)$ denote the feasible candidate-task set in DS d. A GP individual h acts as a scoring rule that assigns a priority value to each candidate task based on its feature vector $\mathbf{x}(d, t)$. In the current implementation, smaller scores indicate higher priority, and the candidate receiving the lowest score under h

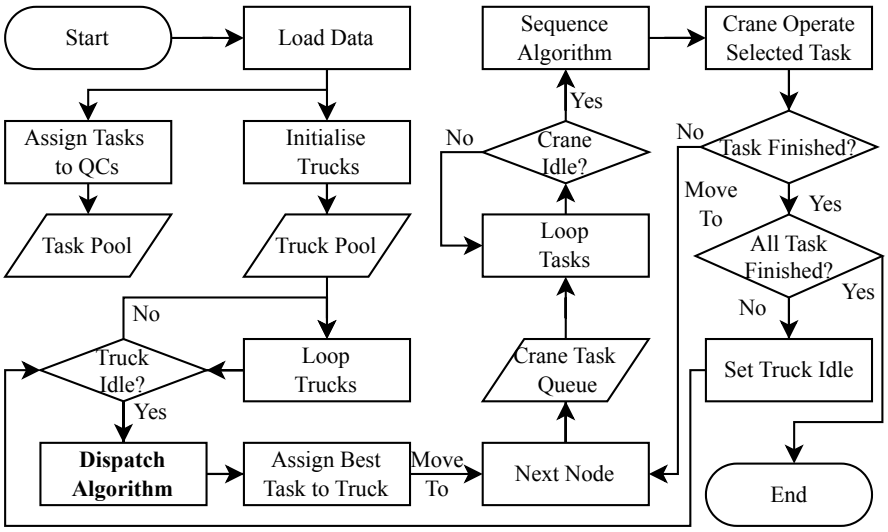


Fig. 1. Event-driven dispatch workflow of the DCTTS simulator, adapted from [19].

is selected. Repeating this procedure throughout the simulation run induces a complete ranking over the candidate tasks in each DS, which is precisely the behavioral information exploited by the proposed characterization.

3 Weighted Spearman Phenotypic Characterization

WSPC characterizes each GP individual by comparing the complete ranking it induces in each selected decision situation (DS) with that of a fixed reference policy. This reference-based design yields a fixed-length embedding over the sampled DS set while avoiding the quadratic cost of pairwise rank comparisons across individuals. In addition, since candidate-set sizes vary across DSs, it provides a more comparable basis for behavioral comparison than naive encodings based on task identities or raw rank positions.

To better reflect the importance of early decisions in dynamic scheduling, WSPC further introduces position-dependent weights into the rank comparison. As a result, disagreements near the top of the ranking receive greater emphasis, while lower-ranked tasks remain represented. This yields a richer behavioral characterization than conventional top-1 PC.

3.1 Formulation

Since rank correlation is meaningful only when at least two candidates are available, we only use DSs with $n_i \geq 2$. Singleton candidate sets are discarded because they contain no ranking information. Let $\mathcal{D} = \{d_i\}_{i=1}^{M}$ denote the selected

set of M decision situations (DSs). For each $d_i \in \mathcal{D}$, let $\Omega(d_i) = [t_{i,u}]_{u=1}^{n_i}$ denote the candidate list presented by the simulator at d_i, where u is the deterministic list index and n_i is the number of candidates at d_i (which may vary across DSs). Given GP individual h and fixed reference rule h^{ref}, both assign a real-valued priority score $\pi_i(\cdot, t) \in \mathbb{R}$ to each $t \in \Omega(d_i)$. In the current study, h^{ref} is instantiated as the deterministic manual baseline [4, 19].

In our simulator, smaller priority scores indicate higher preferences. Accordingly, the rank of each candidate is obtained by sorting $\Omega(d_i)$ in ascending order of $\pi_i(\cdot, t)$, with ties broken deterministically by the index u. This yields unique ranks under both h and h^{ref}. For each $t_{i,u} \in \Omega(d_i)$, let $r_i(h, t_{i,u}) \in \{1, \ldots, n_i\}$ denote its rank under h (1 indicates the highest priority), and let $r_i(h^{\mathrm{ref}}, t_{i,u}) \in \{1, \ldots, n_i\}$ denote the corresponding rank under h^{ref}.

To emphasize discrepancies near the top of the ranking without discarding lower-ranked tasks, we use the following monotone discount inspired by discounted cumulative gain (DCG) [8]:

$$g(k) = \frac{1}{\log_2(1+k)}, \qquad k = 1, \ldots, n_i. \tag{1}$$

We then assign each candidate a normalized weight based on its reference rank:

$$w_i(u) = \frac{g\big(r_i(h^{\mathrm{ref}}, t_{i,u})\big)}{\sum_{j=1}^{n_i} g(r_i(h^{\mathrm{ref}}, t_{i,j}))}, \qquad u = 1, \ldots, n_i. \tag{2}$$

By construction, $w_i(u) \geq 0$ and $\sum_{u=1}^{n_i} w_i(u) = 1$, so all candidates remain represented while higher-priority reference choices receive larger weights. For each d_i, the reference ranks $\{r_i(h^{\mathrm{ref}}, t_{i,u})\}_{u=1}^{n_i}$ and weights $\{w_i(u)\}_{u=1}^{n_i}$ depend only on d_i and h^{ref}, and can therefore be precomputed and reused across individuals.

Using the precomputed weights $\{w_i(u)\}_{u=1}^{n_i}$, we compute the weighted Spearman correlation [18] between h and h^{ref} in each $d_i \in \mathcal{D}$ via weighted Pearson correlation on the induced rank lists. For each $t_{i,u} \in \Omega(d_i)$ $(u = 1, \ldots, n_i)$, define

$$r_{i,u}(h) = r_i(h, t_{i,u}), \qquad r_{i,u}^{\mathrm{ref}} = r_i(h^{\mathrm{ref}}, t_{i,u}). \tag{3}$$

The corresponding weighted means are

$$\bar{r}_i(h) = \sum_{u=1}^{n_i} w_i(u)\, r_{i,u}(h), \qquad \bar{r}_i^{\mathrm{ref}} = \sum_{u=1}^{n_i} w_i(u)\, r_{i,u}^{\mathrm{ref}}. \tag{4}$$

Define the weighted covariance and variances

$$\begin{aligned} S_i^{h,\mathrm{ref}}(h) &= \sum_{u=1}^{n_i} w_i(u)\big(r_{i,u}(h) - \bar{r}_i(h)\big)\big(r_{i,u}^{\mathrm{ref}} - \bar{r}_i^{\mathrm{ref}}\big), \\ S_i^{h,h}(h) &= \sum_{u=1}^{n_i} w_i(u)\big(r_{i,u}(h) - \bar{r}_i(h)\big)^2, \\ S_i^{\mathrm{ref},\mathrm{ref}} &= \sum_{u=1}^{n_i} w_i(u)\big(r_{i,u}^{\mathrm{ref}} - \bar{r}_i^{\mathrm{ref}}\big)^2. \end{aligned} \tag{5}$$

The weighted Spearman correlation in d_i is then

$$\rho_i(h) = \frac{S_i^{h,\mathrm{ref}}(h)}{\sqrt{S_i^{h,h}(h)\, S_i^{\mathrm{ref},\mathrm{ref}}}}. \tag{6}$$

To obtain a normalized dissimilarity in $[0, 1]$, we map

$$p_i(h) = \frac{1 - \rho_i(h)}{2} \in [0, 1]. \tag{7}$$

Finally, the WSPC vector of h over all selected DSs is

$$\mathbf{p}(h) = \big[p_1(h), \ldots, p_M(h)\big] \in [0, 1]^M, \tag{8}$$

which serves as a scale-consistent behavioral embedding. This vector can be used directly for distance computation (e.g., Euclidean distance) in surrogate modeling and behavior-space grouping.

As a simple illustration, consider one DS with four candidates A, B, C, D and reference order $A \succ B \succ C \succ D$. Suppose two rules both rank A first. Conventional PC assigns the same top-1 characterization to both rules. If h_1 induces the same ranks $(1, 2, 3, 4)$ as the reference, then WSPC gives $p_i(h_1) = 0$. If h_2 induces ranks $(1, 4, 3, 2)$, the two rules still agree on the top candidate, but WSPC captures the remaining rank disagreement. Using the DCG weights in (2), approximately $(0.3904, 0.2463, 0.1952, 0.1681)$, gives $\rho_i(h_2) = 0.4013$ and $p_i(h_2) = 0.2994$ by (7). Thus WSPC separates rules that PC treats as identical.

3.2 Computational Complexity

For each selected $d_i \in \mathcal{D}$, we compute the reference ranks $\{r_i(h^{\mathrm{ref}}, t_{i,u})\}_{u=1}^{n_i}$ and the corresponding weights $\{w_i(u)\}_{u=1}^{n_i}$ once and reuse them across all individuals. Both this precomputation step and the ranking step required to compute WSPC for one individual are dominated by sorting the candidate list in each decision situation, yielding a complexity of $O\Big(\sum_{i=1}^{M} n_i \log n_i\Big)$. For a population of size N, the per-generation cost scales linearly with N.

4 Experimental Design

4.1 Simulation Environment and Datasets

All methods are evaluated using the DCTTS simulator described in Section 2.3. We construct eight scenarios using the same instance generator to cover varied operating conditions. Each scenario is indexed by three factors: the number of tasks per instance, which is either 200 or 400, the proportion of loading tasks, which is either 0.3 or 0.7, and the fleet size, which is either 5 or 8 trucks per quay crane. The eight scenarios are formed by all combinations of these factors, and each scenario is denoted by ⟨number of tasks, loading-task proportion, trucks per quay crane⟩. For each scenario, we generate 10 training instances and 100 testing instances, giving 80 training and 800 testing instances in total.

4.2 Fitness Evaluation

We adopt an average relative deviation (ARD)-based fitness definition with respect to a fixed reference policy, with a small bloat penalty [1, 19]:

$$F(h) = \frac{1}{|\mathcal{I}_{\text{train}}|} \sum_{m \in \mathcal{I}_{\text{train}}} \frac{\text{Obj}(m,h) - \text{Obj}(m,\text{ref})}{\text{Obj}(m,\text{ref})} - \lambda \cdot S(h), \tag{9}$$

where $\mathcal{I}_{\text{train}}$ is the training-instance set, $\text{Obj}(m,h)$ is the throughput of rule h on instance m, ref is the manual baseline, $S(h)$ is tree size, and λ is the penalty factor. Larger values of $F(h)$ indicate better performance.

4.3 Comparison Protocol

To isolate the effect of characterization design, the main experiments use the same pre-selection surrogate-assisted GP framework as the PC baseline [7], with a fixed 1NN surrogate as the base model. Thus, the compared SGP-PS variants differ only in their characterizations. We compare the following methods:

- **GP**: standard genetic programming without surrogate assistance.
- **SGP-PS(PC)**: pre-selection SAGP using conventional top-1 phenotypic characterization [7].
- **SGP-PS(SPC)**: pre-selection SAGP using unweighted full-ranking Spearman characterization.
- **SGP-PS(WSPC)**: pre-selection SAGP using weighted Spearman phenotypic characterization with DCG weighting.

To ensure a fair comparison among PC, SPC, and WSPC, we collect a pool of decision situations (DSs) from the training instances using a committee comprising the manual baseline and eight simple dispatching rules. These rules select tasks based on the shortest or longest travel time, the smallest or largest number of remaining tasks, the lowest or highest queue pressure, and the shortest or longest service time. The committee is used only for DS collection, while the manual baseline is retained as the fixed reference policy for all characterizations. We discard DSs with fewer than two feasible candidates and remove exact duplicates. From the remaining pool, we select $|\mathcal{D}| = 40$ DSs by candidate-set-size stratification, sampling two or three DSs for each size from 2 to 16 to provide balanced coverage of candidate-set sizes. The simulator-provided candidate order is retained, and the same selected DSs are used for PC, SPC, and WSPC. All methods are evaluated over 30 runs using the same set of random seeds, and pairwise differences are assessed using the Wilcoxon signed-rank test at $\alpha = 0.05$.

4.4 Parameter Settings

We adopt a fixed 18,000 s training-time budget so that all compared methods are evaluated under the same computational budget. All common GP parameters are kept fixed across scenarios, methods, and random seeds. The pre-selection brood size is 2. Following [19], the population size is 500, tournament size is 5, the number of elites is 10, the crossover/mutation/reproduction rates are 0.8/0.15/0.05, the maximum tree depth is 10, and the penalty factor is 10^{-7}.

Table 1. Mean (std) test performance (×100) over 30 runs.

Scenario	GP	SGP-PS(PC)	SGP-PS(SPC)	SGP-PS(WSPC)
Overall	6.30(0.31)(+)	6.54(0.33)(+)	6.61(0.33)(+)	**6.71(0.22)**
$\langle 200, 0.3, 5\rangle$	7.35(0.60)(+)	7.84(0.71)(+)	7.88(0.67)(+)	**8.28(0.65)**
$\langle 200, 0.3, 8\rangle$	1.88(0.38)(+)	2.21(0.42)(=)	2.21(0.46)(=)	**2.31(0.31)**
$\langle 200, 0.7, 5\rangle$	6.47(0.23)(+)	6.59(0.36)(=)	6.58(0.33)(=)	**6.60(0.31)**
$\langle 200, 0.7, 8\rangle$	4.62(0.25)(=)	**4.69(0.25)**(=)	4.65(0.22)(=)	4.66(0.20)
$\langle 400, 0.3, 5\rangle$	10.94(0.97)(+)	11.58(1.02)(+)	11.83(0.94)(+)	**12.29(0.64)**
$\langle 400, 0.3, 8\rangle$	4.20(0.49)(+)	4.47(0.35)(+)	4.46(0.37)(+)	**4.68(0.30)**
$\langle 400, 0.7, 5\rangle$	7.76(1.07)(=)	7.71(1.17)(=)	**8.01(1.20)**(=)	7.67(1.47)
$\langle 400, 0.7, 8\rangle$	7.22(0.22)(=)	7.22(0.28)(=)	**7.28(0.22)**(=)	7.22(0.25)

Note: Higher is better. Symbols compare SGP-PS(WSPC) with each non-WSPC method using the Wilcoxon signed-rank test at $\alpha = 0.05$: (+) better, (−) worse, and (=) no significant difference.

5 Results and Discussion

5.1 Scheduling Performance

Table 1 reports the overall and scenario-level test performance under the same training-time budget. All three SGP-PS variants improve the overall mean performance over standard GP, and SGP-PS(WSPC) achieves the best overall result. Its improvements over GP, SGP-PS(PC), and SGP-PS(SPC) are statistically significant under the Wilcoxon signed-rank test ($p < 0.001$, $p = 0.019$, and $p = 0.045$, respectively).

At the scenario level, SGP-PS(WSPC) obtains the highest mean performance in five of the eight scenarios. It is best in all four scenarios with a loading-task proportion of 0.3, with particularly pronounced gains in $\langle 200, 0.3, 5\rangle$ and $\langle 400, 0.3, 5\rangle$. When the loading-task proportion is 0.7, the differences among the SGP-PS variants are not statistically significant. SGP-PS(PC) has the highest mean in $\langle 200, 0.7, 8\rangle$, while SGP-PS(SPC) is best in both 400-task scenarios.

These results are consistent with the information retained by the three characterizations. PC uses only the top-ranked candidate and can miss differences in the remaining preference order. SPC preserves the complete ranking, but its uniform treatment of all rank positions may make the representation sensitive to less consequential tail-ranking differences. WSPC refines this full-ranking signal by emphasizing candidates that are highly ranked by the reference policy, which is appropriate for DCTTS because the dispatching rule executes only the highest-priority task at each decision point. This more decision-relevant representation can provide more effective surrogate guidance, especially when the rule has greater freedom to alter task choices. Loading tasks follow a prescribed within-crane order, whereas discharge tasks offer greater scheduling flexibility. The lower loading-task proportion of 0.3 therefore provides more scope for WSPC's surrogate guidance

to translate into scheduling improvements, offering a plausible explanation for the more pronounced gains in these scenarios. The results suggest that the weighted full-ranking representation provides the most effective characterization among the compared SGP-PS variants.

Table 2. Time-weighted mean (std) surrogate quality of SGP-PS variants over 30 runs.

Metric	SGP-PS(PC)	SGP-PS(SPC)	SGP-PS(WSPC)
Spearman	0.7577(0.0409)(+)	**0.8088(0.0486)**(=)	**0.8088(0.0489)**
Top-10% hit	0.3933(0.0822)(+)	0.5267(0.0837)(=)	**0.5390(0.0664)**
Top-20% hit	0.5388(0.0852)(+)	0.6423(0.0786)(=)	**0.6695(0.0731)**

Note: Higher is better. Symbols follow Table 1.

5.2 Surrogate Quality

To further assess how characterization affects surrogate learning, Table 2 reports diagnostic surrogate-quality results under the same training-time budget. Each metric is averaged over the 18,000 s horizon according to the amount of training time covered by each observed value. Spearman correlation evaluates global rank preservation, while top-q% hit is the fraction of true-fitness top-q% individuals also placed in the surrogate-predicted top q%, directly reflecting pre-selection quality. SGP-PS(WSPC) significantly outperforms SGP-PS(PC) on all reported metrics. The SPC and WSPC variants achieve virtually identical Spearman correlations, indicating comparable global ranking quality, while WSPC attains the highest average top-10% and top-20% hit rates. This pattern is consistent with WSPC's focus on decision-relevant ranking differences. As pre-selection is applied repeatedly throughout evolution, selecting even one additional promising offspring for true evaluation may alter the population and subsequent search trajectory. Small improvements in top-k identification may therefore accumulate across generations, providing a plausible link between WSPC's surrogate guidance and its final scheduling performance.

6 Conclusion

This paper proposes weighted Spearman phenotypic characterization (WSPC) for surrogate-assisted genetic programming in dynamic scheduling. By replacing top-1 characterization with a full-ranking, reference-based, and position-aware behavioral embedding, WSPC preserves richer behavioral information while remaining compatible with existing distance-based surrogate models.

The DCTTS experiments show that WSPC achieves the best average scheduling performance across the eight tested scenarios under the adopted pre-selection SAGP framework. It significantly outperforms standard GP, SGP-PS(PC), and

SGP-PS(SPC). The surrogate-quality diagnostics show that WSPC significantly improves all reported metrics over PC and achieves the highest average top-k hit rates, supporting its ability to identify promising individuals during pre-selection. These results suggest that full-ranking phenotypic characterizations are more effective than conventional top-1 PC, and that position-aware weighting can further refine full-ranking representations.

Future work can evaluate WSPC with other distance-based surrogate models and other dynamic scheduling problems, and examine how alternative weighting schemes, reference-policy choice, and decision-situation sampling affect the robustness of phenotypic characterizations.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 72071116, and in part by the Ningbo Science and Technology Bureau under Grants 2023Z237 and 2025Z197. We are grateful for support from the Digital Port Technologies Lab and access to the High Performance Computing Facility at the University of Nottingham Ningbo China.

References

1. Branke, J., Nguyen, S., Pickardt, C.W., Zhang, M.: Automated design of production scheduling heuristics: A review. IEEE Transactions on Evolutionary Computation **20**(1), 110–124 (2016). https://doi.org/10.1109/TEVC.2015.2429314
2. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R.: Hyper-heuristics: A survey of the state of the art. Journal of the Operational Research Society **64**(12), 1695–1724 (2013). https://doi.org/10.1057/jors.2013.71
3. Chen, X., Bai, R., Qu, R., Dong, H.: Cooperative double-layer genetic programming hyper-heuristic for online container terminal truck dispatching. IEEE Transactions on Evolutionary Computation **27**(5), 1220–1234 (2023). https://doi.org/10.1109/TEVC.2022.3209985
4. Chen, X., Bai, R., Qu, R., Dong, H., Chen, J.: A data-driven genetic programming heuristic for real-world dynamic seaport container terminal truck dispatching. In: 2020 IEEE Congress on Evolutionary Computation (CEC). pp. 1–8. IEEE (2020). https://doi.org/10.1109/CEC48606.2020.9185659
5. Chen, X., Bai, R., Qu, R., Dong, J., Jin, Y.: Deep reinforcement learning assisted genetic programming ensemble hyper-heuristics for dynamic scheduling of container port trucks. IEEE Transactions on Evolutionary Computation **29**(4), 1371–1385 (2025). https://doi.org/10.1109/TEVC.2024.3381042
6. Forrester, A.I.J., Keane, A.J.: Recent advances in surrogate-based optimization. Progress in Aerospace Sciences **45**(1–3), 50–79 (2009). https://doi.org/10.1016/j.paerosci.2008.11.001
7. Hildebrandt, T., Branke, J.: On using surrogates with genetic programming. Evolutionary Computation **23**(3), 343–367 (2015). https://doi.org/10.1162/EVCO_a_00133
8. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems **20**(4), 422–446 (2002). https://doi.org/10.1145/582415.582418

9. Jin, C., Bai, R., Zhou, Y., Chen, X., Tan, L.: Enhancing online yard crane scheduling through a two-stage rollout memetic genetic programming. Memetic Computing **16**(3), 467–489 (2024). https://doi.org/10.1007/s12293-024-00424-4
10. Jin, J., Cui, T., Bai, R., Qu, R.: Container port truck dispatching optimization using Real2Sim based deep reinforcement learning. European Journal of Operational Research **315**(1), 161–175 (2024). https://doi.org/10.1016/j.ejor.2023.11.038
11. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing **9**(1), 3–12 (2005). https://doi.org/10.1007/s00500-003-0328-5
12. Jin, Y.: Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation **1**(2), 61–70 (2011). https://doi.org/10.1016/j.swevo.2011.05.001
13. Koza, J.R.: Genetic programming as a means for programming computers by natural selection. Statistics and Computing **4**(2), 87–112 (1994). https://doi.org/10.1007/BF00175355
14. Moraglio, A., Poli, R.: Geometric landscape of homologous crossover for syntactic trees. In: 2005 IEEE Congress on Evolutionary Computation. vol. 1, pp. 427–434. IEEE (2005). https://doi.org/10.1109/CEC.2005.1554715
15. Nguyen, S., Mei, Y., Zhang, M.: Genetic programming for production scheduling: A survey with a unified framework. Complex & Intelligent Systems **3**(1), 41–66 (2017). https://doi.org/10.1007/s40747-017-0036-x
16. Pawlik, M., Augsten, N.: Efficient computation of the tree edit distance. ACM Transactions on Database Systems **40**(1), 3:1–3:40 (2015). https://doi.org/10.1145/2699485
17. Pawlik, M., Augsten, N.: Tree edit distance: Robust and memory-efficient. Information Systems **56**, 157–173 (2016). https://doi.org/10.1016/j.is.2015.08.004
18. Spearman, C.: The proof and measurement of association between two things. The American Journal of Psychology **15**(1), 72–101 (1904). https://doi.org/10.2307/1412159
19. Tan, L., Jin, C., Chen, X., Qu, R., Bai, R.: PGU-SGP: A pheno-geno unified surrogate genetic programming for real-life container terminal truck scheduling. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 322–330. GECCO '25, ACM (2025). https://doi.org/10.1145/3712256.3726326
20. Vanneschi, L., Castelli, M., Silva, S.: A survey of semantic methods in genetic programming. Genetic Programming and Evolvable Machines **15**(2), 195–214 (2014). https://doi.org/10.1007/s10710-013-9210-0
21. Yang, X., Li, Y., Cui, T., Bai, R.: A collaborative evolutionary reinforcement learning approach to dynamic pickup and delivery challenges. Transportation Research Part E: Logistics and Transportation Review **212**, 104885 (2026). https://doi.org/10.1016/j.tre.2026.104885
22. Zhang, Y., Bai, R., Qu, R., Tu, C., Jin, J.: A deep reinforcement learning based hyper-heuristic for combinatorial optimisation with uncertainties. European Journal of Operational Research **300**(2), 418–427 (2022). https://doi.org/10.1016/j.ejor.2021.10.032
23. Zhu, L., Zhang, F., Zhu, X., Chen, K., Zhang, M.: Sample-aware surrogate-assisted genetic programming for scheduling heuristics learning in dynamic flexible job shop scheduling. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 384–392. ACM (2023). https://doi.org/10.1145/3583131.3590440
24. Zhu, L., Zhang, F., Zhu, X., Chen, K., Zhang, M.: Phenotype and genotype based sample aware surrogate-assisted genetic programming in dynamic flexible job shop scheduling. IEEE Transactions on Artificial Intelligence **6**(12), 3232–3247 (2025). https://doi.org/10.1109/TAI.2025.3562161

Constraint Learning and Schedule Proposal via Transformers for the Nurse Scheduling Problem

Walid Abdelaidoum[1,2], Mohamed A. Madani[2], Larbi Boubchir[1], and Boubaker Daachi[1]

[1] LIASD Laboratory, University of Paris 8, Paris, France
[2] SWAPPY, Paris, France

Abstract. We train a transformer on feasible schedules of a simplified Nurse Scheduling Problem variant generated by a constraint programming solver, and use the trained model as a learnability probe with no constraint fed at inference. We evaluate three constraint configurations spanning 3 to 40 nurses and report per-family satisfaction rates across four constraint families. The results reveal a learnability hierarchy by structural scope: local per-nurse rules are captured reliably, bounded-window rules degrade with sequence length, and global cross-nurse rules remain the hardest to satisfy. Median violation counts stay low (0 to 5.5 per schedule over the full horizon), suggesting that the generated schedules are suitable as warm starts for repair or local-search procedures.

Keywords: Nurse Scheduling Problem · Transformers · Constraint Programming · Learnability

1 Introduction

The Nurse Scheduling Problem (NSP) assigns shifts to nurses over a planning horizon subject to hard and soft constraints. We study a simplified variant with hard constraints only and four shift types Night (N), Day (D), Evening (E), and Rest (R), over a horizon of d days for a team of $|\mathcal{I}|$ nurses. Many practically relevant NSP variants are NP-hard, although some sub-classes are polynomially solvable [3].

Recent NSP work falls in three lines. Exact and decomposition approaches model the problem as mixed-integer programs solved monolithically or by branch-and-price [2, 8]. Metaheuristic and matheuristic methods address larger instances, e.g., simulated annealing on real rosters [4] and hybrid meta-heuristics on extended formulations with reserve nurses [9]. Neural networks have been introduced inside NSP solvers to guide or assist an existing search procedure, through recurrent-network-assisted heuristics [6], deep-network-assisted heuristic selection [5], multi-agent deep Q-network-based neighbourhood selection [11], or federated LSTM-based fatigue prediction [1]. Beyond the NSP, decoder-only transformers have been applied to combinatorial problems such as Sudoku, where pairing a small GPT-2 with depth-first search and a verifier reaches near-perfect accuracy [7].

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

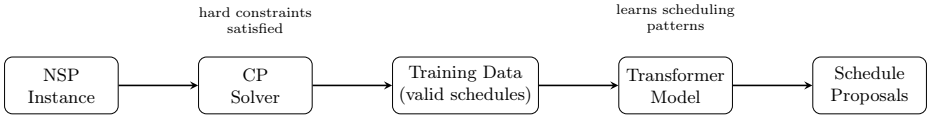


Fig. 1. Pipeline: a CP solver generates feasible schedules; a transformer is trained on them and proposes new ones.

The present work uses a transformer as a learnability probe: no constraint is fed to the model; we measure which constraint families the transformer captures from feasible examples alone. Figure 1 illustrates the proposed pipeline.

2 Problem Formulation and Transformer Architecture

Let $\mathcal{I} = \{0, \ldots, n-1\}$ denote the set of nurses, $\mathcal{J} = \{0, \ldots, d-1\}$ the set of days, and $\mathcal{S} = \{N, D, E, R\}$ the four shift types (Night, Day, Evening, Rest); the decision variable $x_{i,j} \in \mathcal{S}$ gives the shift of nurse i on day j. We further use $\mathcal{F} \subseteq \mathcal{S} \times \mathcal{S}$ for forbidden consecutive pairs, $\mathcal{T} \subseteq \mathcal{S}$ for shifts subject to consecutive-day limits, $\mathcal{M} \subseteq \mathcal{S}$ (with $\mathcal{M} \not\subseteq \mathcal{T}$) for the mandatory shifts following a maximal run, $k_{\max} \in \mathbb{Z}^+$ for the maximum consecutive shifts from $\mathcal{T}$, $p \in \mathbb{Z}^+$ for the rolling-window period, and triples (s, l, u) for coverage requirements (shift s with bounds $[l, u]$); the sliding-window start day is $j_0 \in \{0, \ldots, d-1-k_{\max}\}$ and the indicator $\mathbf{1}[x_{i,j} \in \mathcal{T}]$ marks whether nurse i works a shift in $\mathcal{T}$ on day j. The CP model reads

$$\min \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} \sum_{s \in \mathcal{S}} w_{i,j,s}\, \mathbf{1}[x_{i,j} = s], \tag{1}$$

subject to

$$(x_{i,j},\, x_{i,j+1}) \notin \mathcal{F},\ \forall\, i,\ j \in \{0, \ldots, d-2\}, \tag{2}$$

$$\sum_{\delta=0}^{k_{\max}} \mathbf{1}[x_{i,\, j_0+\delta} \in \mathcal{T}] \le k_{\max},\ \forall\, i, j_0, \tag{3}$$

$$\bigwedge_{\delta=0}^{k_{\max}-1} \left(x_{i,\, j_0+\delta} \in \mathcal{T}\right) \ \Rightarrow\ x_{i,\, j_0+k_{\max}} \in \mathcal{M}, \tag{4}$$

$$l \le \sum_{i \in \mathcal{I}} \mathbf{1}[x_{i,j} = s] \le u, \tag{5}$$

$$l_W \le \sum_{j \in W} \mathbf{1}[x_{i,j} \in \mathcal{A}] \le u_W,\ \forall\, i, W. \tag{6}$$

The objective (1) carries no preference: weights $w_{i,j,s} \sim \mathcal{U}[0, 1]$ are redrawn at each solver call so that independent runs return distinct feasible schedules used as training data. A schedule is feasible if it satisfies the five constraint families (consecutive limits and the mandatory-shift rule are grouped into one evaluation family in Section 3): forbidden transitions on adjacent days, Eq. (2); maximum consecutive shifts from $\mathcal{T}$, Eq. (3); mandatory shift in $\mathcal{M}$ after a maximal run, Eq. (4); daily coverage bounds (applied uniformly to all days, weekends included), Eq. (5); shift-limit bounds over rolling windows of p days, Eq. (6). Table 1 shows the active constraints per data configuration.

Table 1. Active constraints per configuration. Dash: inactive. Parameter values $k_{\max}$, coverage bounds, and $\mathcal{A}_k$ subsets in the text.

	Forb. (2)	Consecutive (3)			Cov. (5)	Per 7 d (6)
	N → D	nts → R	wrk → R	rst → wrk (4)		
S_1	✓	3	5	—	✓	✓
S_2	✓	3	4	2	✓	—
S_3	✓	3	—	—	✓	✓

Transformer architecture: Each schedule is serialised row-by-row into a token sequence over a vocabulary of size $|\mathcal{V}|$=136: the four shift tokens from $\mathcal{S}$, four special tokens BOS/PAD/SEP/EOS, and condition tokens encoding $|\mathcal{I}|$ and $|\mathcal{J}|$. Three learned positional embeddings (sequence, nurse, day) are summed to recover the 2D coordinate that flattening loses; about 65% of model parameters live in these tables. The model is a decoder-only transformer [10] (4 layers × 4 heads, d_{model}=128, ∼837K parameters), Pre-LayerNorm, masked self-attention, GELU (Gaussian Error Linear Unit) feed-forward, output projection weight-tied to the token embedding. Training minimises masked cross-entropy with AdamW, learning rate 3×10^{-4}, linear warmup then cosine decay, gradient clipping at norm 1, and early stopping after 5 epochs without validation improvement. At inference, the trained transformer generates a complete assignment $x_{i,j}$ autoregressively by greedy left-to-right decoding; no CP model is used.

3 Experimental Evaluation

Each trained model generates 150 schedules on held-out instance dimensions; the constraint checker tests each of the four families independently (consecutive limits and the mandatory-shift rule count as one family). The main metric is the satisfaction rate, the fraction of schedules that fully respect a given constraint.

All three models converge with training-loss plateaus of 0.78 (S_1), 0.69 (S_2), and 0.94 (S_3), reflecting the entropy of the feasible-solution space.

Constraint satisfaction rates are: forbidden transitions 80.8–99.3%, consecutive limits 30.7–100%, coverage 14.0–86.7%, and period limits 7.3–62.0%, exhibiting a clear hierarchy local > bounded-window > global cross-nurse.

Median total violations per schedule are 0 (S_2), 4 (S_3), and 5.5 (S_1). Even when satisfaction is low, the generated schedules carry few violations and could plausibly seed a repair heuristic or local-search phase rather than being discarded.

A finer breakdown shows where these violations concentrate: on S_1, in night/day coverage (medians ∼0.8–0.9), consecutive work limits (median 1), and period limits (nights/rests per 7 d, medians 2–3); on S_3, in day-coverage shortfalls (median 2). The contrast between S_2 (86.7%) and S_3 (33.3%) on coverage, at identical team sizes, suggests that the added period limits in S_3 compete with coverage targets rather than reinforce them. These failure modes match the warm-start scenario: residual errors lie in a few cross-nurse cells that a repair or local-search step can resolve cheaply.

4 Conclusion

The results reveal a learnability hierarchy by structural scope: local per-nurse rules are captured reliably, bounded-window rules degrade with sequence length, and global

cross-nurse rules remain the hardest. Low median violation counts (0 to 5.5 per schedule over the full horizon) suggest that the model can be used as a warm-start generator for downstream repair or local-search procedures. The study is limited to hard constraints on synthetic instances; per-nurse contracts and soft preferences are left for future work. Logit masking at inference could enforce hard constraints directly.

References

[1] Aibin, M., Chen, F., Menzi, M., Scholte, L., Jiang, Z., Chen, Y., Jin, X., Liu, C., Aibin, A.: Federated learning framework for nurse scheduling to lower fatigue levels of nurses. In: 2025 International Conference on Computing, Networking and Communications (ICNC). pp. 340–344. IEEE (2025)

[2] Böðvarsdóttir, E.B., Bagger, N.C.F., Høffner, L.E., Stidsen, T.J.: A flexible mixed integer programming-based system for real-world nurse rostering. Journal of Scheduling **25**(1), 59–88 (2022)

[3] Burke, E.K., De Causmaecker, P., Berghe, G.V., Van Landeghem, H.: The state of the art of nurse rostering. Journal of scheduling **7**(6), 441–499 (2004)

[4] Ceschia, S., Di Gaspero, L., Mazzaracchio, V., Policante, G., Schaerf, A.: Solving a real-world nurse rostering problem by simulated annealing. Operations Research for Health Care **36**, 100379 (2023)

[5] Chen, Z., De Causmaecker, P., Dou, Y.: A combined mixed integer programming and deep neural network-assisted heuristics algorithm for the nurse rostering problem. Applied Soft Computing **136**, 109919 (2023)

[6] Chen, Z., Dou, Y., De Causmaecker, P.: Neural networked-assisted method for the nurse rostering problem. Computers & Industrial Engineering **171**, 108430 (2022)

[7] Giannoulis, P., Pantis, Y., Tzamos, C.: Teaching transformers to solve combinatorial problems through efficient trial & error. Advances in Neural Information Processing Systems **38**, 133548–133580 (2026)

[8] Legrain, A., Omer, J.: A dedicated pricing algorithm to solve a large family of nurse scheduling problems with branch-and-price. INFORMS Journal on Computing **36**(4), 1108–1128 (2024)

[9] Saemi, S.: Nurse scheduling problem by considering reserve nurses: A mathematical modeling and hybrid meta-heuristic algorithm. Operational Research **25**(4), 103 (2025)

[10] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)

[11] Zhang, X., Yang, Y., Zhu, Q., Lin, Q., Chen, W., Li, J., Coello, C.A.C.: Multiagent deep q-network-based metaheuristic algorithm for nurse rostering problem. Swarm and Evolutionary Computation **87**, 101547 (2024)

Data-Driven Merge-Point Selection in Dynamic-Aircraft-Arrival-Routes Optimization*

Roghayeh Hajizadeh[1][0000−0002−7402−6292], Maksym Moroz[2][0009−0007−5392−7466], Tatiana Polishchuk[2][0000−0002−9869−5992], Elina Rönnberg[1][0000−0002−2081−2888], and Christiane Schmidt[2][0000−0003−2548−5756]

[1] Department of Mathematics (MAI), Linköping University, Sweden
[2] Department of Science and Technology (ITN), Linköping University, Sweden
firstname.lastname@liu.se

Abstract. With dynamic arrival routes (DARs) we can route and schedule aircraft arriving at an airport. DARs entail an arrival tree that merges all arriving traffic during a given time period towards the runway, such that we can guarantee temporal separation and optimal descent profiles for all the arriving aircraft. In this paper, we aim to reduce the complexity of the resulting arrival trees. With DARs, we aim to answer the need for reduced environmental impact while alleviating air-traffic-controller workload through the automated separation. While DARs provide a high degree of flexibility—they allow to accommodate even high-traffic scenarios without reducing capacity—the points in which the entry-point–runway paths merge along the tree, so-called merge points, can be located differently in the terminal maneuvering area (TMA) during different time periods. However, switching these merge points often yields high complexity for air traffic controllers. In this paper, we propose a data-driven approach to select a candidate set of merge points, integrate the restriction to this candidate set into the mathematical framework for DARs, and demonstrate our approach on an example of a realistic high-traffic scenario at Stockholm Arlanda airport. We are able to obtain feasible solutions even for very few merge-point candidates, with a tradeoff of solution quality/performance in the TMA and the complexity induced by the number of merge-point candidates.

Keywords: Dynamic aircraft arrival routes · Integer programming · Data-driven · Candidate merge points.

1 Introduction

A central problem in aviation is to reduce the environmental impact [1, 8], especially with the projected growth in air travel demand (with an average annual growth rate of 4.3% [7]). While this goal is dominant in air traffic in general, a

* This research was supported by grant 2022-03178 (OPT@ATM: Optimization Methods for Large Air Traffic Management Problems) from the Swedish Research Council and by the Swedish Transport Administration via the Friendly TMA project.

specific focus is on terminal maneuvering areas (TMAs), where all traffic converges: environmentally friendly, fuel- and noise-optimized descent profiles can curtail emissions and noise, however, they are not supported by today's strategic standard arrival routes. Hence, air traffic controllers (ATCOs) may increase separation buffers, which decreases throughput, or issue instructions to deviate from optimal trajectories, which maintains throughput but does not leverage the environmental benefits. Any solution must also take the work of the ATCOs into account: the increasing traffic raises ATCOs' workload and the complexity of their task, possibly impacting safety.

A promising way to reduce both complexity and workload is to design arrival-route structures that support automated separation while keeping high throughput. Fixed topologies such as point merge (PM) [3] and trombone procedures [15] have been studied as means to support path stretching and shortcutting. PM is expected to improve both controller workload and environmental performance [2, 3]. For PM, Hardell et al. [6] showed that all arrivals can follow the most fuel-efficient continuous descent operations (CDOs) while improving time and distance in TMA. Sáez et al. [12] provided a similar analysis for trombones and showed that, for Frankfurt am Main airport, all aircraft under low traffic but not under high traffic can be scheduled with CDOs. Liu et al. [9] showed PM outperforms trombones in capacity and procedure. These concepts, however, rely on fixed structures and therefore offer limited flexibility.

For this reason, we have studied dynamic arrival routes (DARs), a route structure in which aircraft from different entry points are guided along paths. The entry-point–runway paths are required to form an arrival tree: each entry point has exactly one path to the runway, and whenever two paths merge, they remain merged all the way to the runway. Intuitively, the structure looks like a tree that starts at the runway (root) and branches out toward the entry points (leaves). Figure 1 shows an example of an arrival tree for Stockholm Arlanda airport, where traffic from the airport's four entry points is merged toward the runway. The tree structure allows the route geometry to adapt to changing traffic patterns while still maintaining automated separation. The dynamic aspect stems from the fact that the paths constituting the tree may change from one time period to another, resulting in different arrival trees for different traffic situations. An operational concept for DARs was proposed by Sáez et al. [16], and our most recent mathematical model based on a Dantzig–Wolfe reformulation was presented in [4, 5]. In contrast to PM and trombone procedures, DARs can occupy any part of the TMA and therefore better adapt to varying traffic situations. However, we pay for those dynamics with reduced predictability and increased complexity: traffic from different entry points can merge at very different points at different times.

In this paper, we reduce that complexity by restricting merge points to a candidate set—a suggestion from operational ATCOs. This induces three questions: How to identify candidate merge points, how to integrate them into the mathematical model, and what performance impact they have? We answer the first question with a proof-of-concept data-driven approach, the second by a small

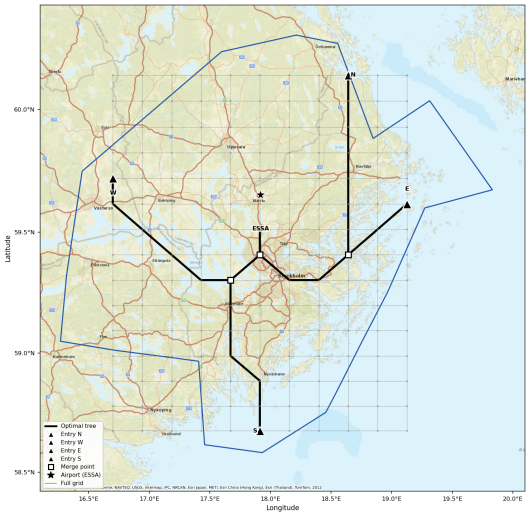


Fig. 1: An arrival tree for Stockholm Arlanda airport

modification of the model, and the third by computing DARs for Stockholm Arlanda airport and studying how the number of candidates affects the results.

The remainder of the paper is organized as follows. Section 2 defines the problem, Section 3 presents the optimization framework, Section 4 proposes the data-driven candidate selection approach, Section 5 presents the experiments, and Section 6 concludes the paper.

2 Problem Formulation

We are given the locations of TMA entry points and the runway, a set of aircraft planned for arrival within a given time interval, the planned arrival time at the entry point for each aircraft, path-dependent descent profiles for these aircraft, and separation requirements. We aim to compute:

A. An optimal arrival tree, such that all aircraft follow the path in the tree from their entry point to the final approach fix, from where they will proceed to the runway. We usually refer to these as entry-point–runway paths.
B. A set of points, $\mathcal{M}$, in the TMA that serves as a candidate set for merge points, where the traffic from the different entry points merges to the runway.

The tree must fulfill various conditions:

1. No more than two routes merge at a point from $\mathcal{M}$ and no routes merge at any point not in $\mathcal{M}$ (to limit complexity for ATCOs).
2. Merge points are separated by a minimum distance D, [11] (to prevent many merge points in close vicinity).
3. Routes do not make sharp turns in order to respect aircraft dynamics.

4. Obstacles, like no-fly zones, are avoided.
5. Temporal separation depending on the wake-turbulence categories of leading and trailing arriving aircraft is ensured.
6. All aircraft arrive to their entry points within a given time interval around the planned time.
7. All aircraft follow a set of descent profiles; in this paper, all aircraft follow optimal descent profiles.

Our tree should be evaluated w.r.t. two criteria: the total distance flown in the TMA (short flight routes for the aircraft), and the weight of the tree (low complexity for the ATCOs and a smaller noise-affected area).

3 Dynamic-Arrival-Routes Framework

In this section, we describe the components of our optimization framework. We review our path-based model from [5] in Subsection 3.1 to make this paper self-contained and in Subsection 3.2, we present our approach to restricting the merge points of the arrival tree for the DARs to a candidate set $\mathcal{M}$.

3.1 The Path-Based Model

In this subsection, we present the path-based model for the case in which the merge points are not restricted to a candidate set—instead, we only require that no more than two routes merge at any point. However, all the other "ingredients" remain the same even after we introduce the candidate set.

Our model is a path-based model using paths in a graph obtained by discretizing the TMA into a square grid of size $q \times n$ with node set V, where the side length of each grid pixel equals the minimum separation distance D. This Selection also ensures that merge points are separated by at least D (Condition 2, Section 2). We snap all entry points and the runway to the grid and consider the bi-directed grid graph on V, in which each node is connected to its eight neighbors with edges in E. We delete all edges that interact with obstacles (Condition 4), all incoming edges of entry points, and all outgoing edges of the runway. Let l_{ij} denote the length of edge $(i,j) \in E$, $\mathcal{P} \subseteq V$ the set of entry points, and $r \in V$ the runway. Finally, let $V' = V \setminus \{\text{last grid row}\} \setminus \{\text{last grid column}\}$.

We further need some notation on the aircraft we aim to schedule. We denote the set of aircraft arriving to entry point $b \in \mathcal{P}$ by $\mathcal{A}_b$, the set of all aircraft by $\mathcal{A} = \bigcup_{b \in \mathcal{P}} \mathcal{A}_b$, and the planned arrival time of aircraft a at its entry point by $\bar{t}_a$. Since these fixed times may cause infeasibility, we allow an aircraft to be scheduled to arrive at any point in an interval $[\bar{t}_a - \mu, \bar{t}_a + \mu]$ for $\mu \geq 0$. Additionally, to limit the time horizon for which the schedule needs to be computed, we denote an upper bound on the time in which all considered aircraft land as $\overline{T}$. We discretize the time and denote the set of time stamps by $T = \{0, \ldots, \overline{T}\}$.

In order to provide automatic separation, let $\mathcal{C} = \{1, 2, \ldots, n_c\}$ be the set of indices of all different disjoint wake-turbulence categories. We denote by C_κ for

$\kappa \in \mathcal{C}$ a set of aircraft from the same category. Hence, we have $\mathcal{A} = \bigcup_{\kappa \in \mathcal{C}} C_\kappa$. We denote the required temporal separation for a leading aircraft from C_{κ_1} and a trailing aircraft from C_{κ_2} by $\sigma_{\kappa_1,\kappa_2}$. Let $\Omega = \max_{\kappa_1,\kappa_2 \in \mathcal{C}} \quad \sigma_{\kappa_1,\kappa_2}$.

In our path-based model, we select a unique path from each entry point to the runway. By restricting how these paths may interact, we enforce Conditions 2–7 and, in the original model, an adapted Condition 1, which only requires that no more than two routes merge at any point. We use one variable per path and therefore generate all feasible paths a priori. Feasibility of a path is determined by the path length, with an upper bound λ on the number of grid-graph edges, and by the angle between consecutive path edges, to ensure Condition 3. We generate the paths with an angle-limited depth-first search from each entry point to the runway, and improve runtime by pruning clearly infeasible branches: we precompute shortest paths, in number of edges, from each grid point to the runway, and cut off a branch at node v whenever the number of edges from the entry point to v plus the precomputed shortest path from v to r exceeds λ.

Since the length of each path is known, we can specify the optimal speed profile of each aircraft on each path. In this paper, we compute descent profiles under international standard atmosphere conditions and no wind for each entry-point–runway path length, assuming neutral CDOs. We employ the method from [16] to determine the optimal vertical profile for each distance to go.

We define some sets based on the paths. We denote the set of paths from entry point b to the runway by Π_b, the set of all paths crossing node $i \in V$ by $\overline{\Pi}_i$, the set of all paths by $\mathbf{\Pi} = \bigcup_{b \in \mathcal{P}} \Pi_b$, and the set of edges that path π passes through for any $\pi \in \mathbf{\Pi}$ by θ_π. Moreover $\xi_{a,\pi,i}$ denotes the time when aircraft a with entry time $\bar{t}_a$ occupies node i on path π based on our pre-computed descent profiles, $\Xi_{a,i,t} = \{\pi \in \overline{\Pi}_i \cap \Pi_b : \quad \xi_{a,\pi,i} = t\}$ denotes the set of all paths on which aircraft a with entry time $\bar{t}_a$ occupies node i at time t, and $\Upsilon_{a,i,t,\sigma} = \{\pi \in \overline{\Pi}_i \cap \Pi_b : \quad t \leq \xi_{a,\pi,i} \leq t + \sigma - 1\}$ denotes the set of all paths on which aircraft a with entry time $\bar{t}_a$ occupies node i sometime between time t and $t + \sigma - 1$. We employ $\Xi_{a,i,t}$ and $\Upsilon_{a,i,t,\sigma}$ to enforce the separation minima (Condition 6). Finally, we introduce binary variables:

- ρ_π indicates whether path π is used in the arrival tree.
- $x_{i,j}$ indicates whether edge $(i,j) \in E$ is used in the arrival tree.
- $\tau_{a,\pi,t}$ indicates whether aircraft a uses path π and arrives at its entry point at time t (to handle flexible entry times).

Using the defined sets and parameters, and introduced variables, we present the path-based formulation, model $\mathrm{M}_{\mathrm{all}}$ given in [5], as:

$$\min \quad \beta \sum_{(i,j)\in E} l_{i,j} x_{i,j} + (1-\beta) \sum_{b \in \mathcal{P}} \sum_{\pi \in \Pi_b} \sum_{(i,j)\in\theta_\pi} |\mathcal{A}_b| l_{i,j} \rho_\pi \tag{1}$$

$$\text{s.t.} \quad \sum_{j:(j,i)\in E} x_{j,i} \leq 2, \qquad \forall i \in V \setminus \{\mathcal{P} \cup r\}, \tag{2}$$

$$\sum_{j:(i,j)\in E} x_{i,j} \leq 1, \qquad \forall i \in V \setminus \{\mathcal{P} \cup r\}, \tag{3}$$

$$x_{i,i+1+n} + x_{i+1+n,i} + x_{i+n,i+1} + x_{i+1,i+n} \leq 1,$$

$$\forall i \in V' \setminus \{\mathcal{P} \cup r\} : i+1+n, i+n, i+1 \notin \{\mathcal{P} \cup r\}, \quad (4)$$

$$x_{i,i+1+n} + x_{i+n,i+1} + x_{i+1,i+n} \leq 1, \qquad \forall i \in \mathcal{P} \cap V', \quad (5)$$

$$x_{i,i+1+n} + x_{i+1+n,i} + x_{i+1,i+n} \leq 1, \qquad \forall i : i+1 \in \mathcal{P}, \quad (6)$$

$$x_{i,i+1+n,} + x_{i+n+1,i} + x_{i+n,i+1} \leq 1, \qquad \forall i : i+n \in \mathcal{P}, \quad (7)$$

$$x_{i+1+n,i} + x_{i+n,i+1} + x_{i+1,i+n} \leq 1, \qquad \forall i : i+n+1 \in \mathcal{P}, \quad (8)$$

$$\sum_{\pi \in \Pi_b} \rho_\pi = 1, \qquad \forall b \in \mathcal{P}, \quad (9)$$

$$\rho_\pi \leq x_{i,j}, \qquad \forall \pi \in \mathbf{\Pi}, \forall (i,j) \in \theta_\pi, \quad (10)$$

$$\sum_{\pi \in \Pi_b} \sum_{t=\bar{t}_a-\mu}^{\bar{t}_a+\mu} \tau_{a,\pi,t} = 1, \qquad \forall b \in \mathcal{P}, \forall a \in \mathcal{A}_b, \quad (11)$$

$$\sum_{t=\bar{t}_a-\mu}^{\bar{t}_a+\mu} \tau_{a,\pi,t} = \rho_\pi, \qquad \forall b \in \mathcal{P}, \forall a \in \mathcal{A}_b, \forall \pi \in \Pi_b, \quad (12)$$

$$\sum_{b \in \mathcal{P}} \sum_{a \in \mathcal{A}_b \cap C_{\kappa_2}} \sum_{t'=-\mu}^{\mu} \sum_{\pi \in \Upsilon_{a,i,t-t',\sigma_{\kappa_1,\kappa_2}}} \tau_{a,\pi,\bar{t}_a+t'} \leq$$

$$\Omega(1 - \sum_{b \in \mathcal{P}} \sum_{a' \in \mathcal{A}_b \cap C_{\kappa_1}} \sum_{t'=-\mu}^{\mu} \sum_{\pi' \in \Xi_{a',i,t-t'}} \tau_{a',\pi',\bar{t}_{a'}+t'}),$$

$$\forall \kappa_1, \kappa_2 \in \mathcal{C} : \kappa_1 \neq \kappa_2,\ \forall i \in V,\ \forall t \in \{0, \ldots, \overline{T} - \sigma_{\kappa_1,\kappa_2} + 1\}, \quad (13)$$

$$\sum_{b \in \mathcal{P}} \sum_{a \in \mathcal{A}_b \cap C_\kappa : a \neq a'} \sum_{t'=-\mu}^{\mu} \sum_{\pi \in \Upsilon_{a,i,t-t',\sigma_{\kappa,\kappa}}} \tau_{a,\pi,\bar{t}_a+t'} \leq \Omega(1 - \sum_{t'=-\mu}^{\mu} \sum_{\pi' \in \Xi_{a',i,t-t'}} \tau_{a',\pi',\bar{t}_{a'}+t'}),$$

$$\forall \kappa \in \mathcal{C}, \forall a' \in C_\kappa, \forall i \in V,\ \forall t \in \{0, \ldots, \overline{T} - \sigma_{\kappa,\kappa} + 1\}, \quad (14)$$

$$x_{i,j} \quad \text{binary}, \qquad \forall (i,j) \in E, \quad (15)$$

$$\rho_\pi \quad \text{binary}, \qquad \forall \pi \in \mathbf{\Pi}, \quad (16)$$

$$\tau_{a,\pi,t} \quad \text{binary}, \qquad \forall b \in \mathcal{P}, \forall a \in \mathcal{A}_b, \forall t \in [\bar{t}_a - \mu, \bar{t}_a + \mu]. \quad (17)$$

The objective function (1) is a convex combination of the paths length and the tree weight. Constraints (2) and (3) ensure that no more than two routes merge at any point—the version without restricting the merge points to the set $\mathcal{M}$. Although degree constraints prevent crossings at nodes, additional constraints such as temporal separations may cause crossings within grid squares, which are prevented by Constraints (4)–(8). On the path level, Constraint (9) ensures that exactly one path is used from each entry point, and Constraint (10) couples the path variables with the selected grid-graph edges. On the aircraft level, Constraints (11) and (12) ensure that exactly one trajectory is chosen for each aircraft. (Constraints (12) and (9) yield Constraint (11), but explicitly including it improved computational performance). Constraints (13) and (14) enforce temporal separations according to the wake-turbulence categories of the leading and trailing aircraft, handling aircraft from different and coinciding categories, respectively. Finally, Constraints (15)–(17) define the domains of variables.

3.2 Integrating the Set of Candidate Merge Points in the Model

As given by Condition 1 for the arrival tree in Section 2, we want to limit the merge points of the tree to nodes from a candidate set $\mathcal{M}$. In model $\mathrm{M_{all}}$, we allowed all grid nodes (except for the entry points and the runway) to be merge

points, by limiting the indegree of any grid node by 2 with Constraint (2). Hence, we can easily integrate the constraint on the set $\mathcal{M}$: We allow at most 2 incoming edges for any node in $\mathcal{M}$ and at most 1 incoming edge for all other nodes. To do so, we substitute Constraint (2) by Constraints (18) and (19).

$$\sum_{j:(j,i)\in E} x_{j,i} \leq 2, \qquad \forall i \in \mathcal{M} \tag{18}$$

$$\sum_{j:(j,i)\in E} x_{j,i} \leq 1, \qquad \forall i \in V \setminus \mathcal{M} \tag{19}$$

The objective function (1), and Constraints (3)–(16), (18)–(19) yield our model for the case that a set of candidate merge points is given—$\mathrm{M}_{\mathrm{cand}}$.

4 Data-Driven Approach to Select the Set of Candidate Merge Points

We propose a data-driven approach to select the set of candidate merge points $\mathcal{M}$, as a proof-of-concept reflecting the data available in practice, to compute (B) in Section 2. We run model $\mathrm{M}_{\mathrm{all}}$ (1)–(17) on a large set of instances $\mathcal{I}$ and track how many times each node appeared as a merge point in the resulting arrival trees. Let m_v denote this number for all $v \in V \setminus \{\mathcal{P} \cup r\}$:

$$m_v = |\{I \in \mathcal{I} |\ v \text{ is merge point in optimal tree for } I\}| \tag{20}$$

We then define $\mathcal{M} := \{v \in V | m_v \geq \bar{m}\}$ for a threshold $\bar{m}$ and use our model $\mathrm{M}_{\mathrm{cand}}$ for this set for deciding on the actual DARs and the underlying tree.

Note that using only a subset of all grid nodes does not guarantee a global optimum for any input instance. Our design selects candidate merge points that (at all or often, for larger values of $\bar{m}$) contributed to global optimal trees for instances from $\mathcal{I}$. If $\mathcal{I}$ with our choice of $\bar{m}$ gives a good representation of the possible traffic situations in the TMA, we often get optimal trees when using these candidates to compute DARs and the underlying arrival tree for a period.

5 Experiments: Stockholm Arlanda Airport

In our experiments, we use data for Stockholm Arlanda obtained from the open-source database of the Opensky Network [10]. The dataset contains 33 aircraft arriving to runway 19L (one of the three runways of Arlanda airport), and the four main entry points HMR (north), XILAN (east), NILUG (south), and ELTOK (west) during the busiest hour of operation in 2018. We use an 11×15 grid, which ensures a merge-point separation of 6 NM. Moreover, we generate all feasible entry-point–runway paths with at most $\lambda = 14$ edges, enforcing a minimum turning angle of 135° between consecutive edges. This results in 3821 feasible paths, computed in less than one second.

Aircraft are classified according to the ICAO wake-turbulence categories [13]: LIGHT (L), MEDIUM (M) and HEAVY (H). In the experiments, we aggregate these categories into two groups, $C_1 = \{H, M\}$ and $C_2 = \{L\}$. Following ICAO separation minima [14], we set the temporal separation parameters as $\sigma_{1,2} = 3$ minutes and $\sigma_{1,1} = \sigma_{2,2} = \sigma_{2,1} = 2$ minutes. Finally, we set $\beta = 0.1$.

In Subsection 5.1, we describe how we obtain a set of instances $\mathcal{I}$ and the resulting set of candidate merge points, and in Subsection 5.2, we present the results for the DARs for different sets of $\mathcal{M}$ based on different values of $\bar{m}$.

5.1 Selection of Merge-Point Candidates

We can generate the set of instances $\mathcal{I}$ either from historic instances or by creating artificial instances from historic data. In this paper, we use the latter to obtain a large variety of arrival trees, based on 33 aircraft from real-world arrivals at Stockholm Arlanda TMA during the busiest hour in 2018: May 16, 5:00–5:59AM.

Algorithm 1: Computation of m_v, $\forall v \in V$.

$m_v = 0$, $\forall v \in V \setminus \{\mathcal{P} \cup r\}$;
$N = |\mathcal{I}|/3$;
for $i = 1, \ldots, N$ **do**
  $k_i \leftarrow$ uniformly at random from $\{7, 8, \ldots, 17\}$;
  Uniformly at random draw a subset $A_i \subset \overline{A}$ with $|A_i| = k_i$;
  for $a \in A_i$ **do**
    $\bar{t}_a \leftarrow$ uniformly at random from $[5{:}30, 5{:}59]$;
  for $\mu = 0, 2, 4$ **do**
    % Input instance I_i^μ is defined;
    Run $\mathrm{M}_{\mathrm{all}}$ on I_i^μ (with parameter μ) to obtain tree T_i^μ;
    if I_i^μ *is feasible* **then**
      for $v \in V \setminus \{\mathcal{P} \cup r\}$ **do**
        if v *is merge point in* T_i^μ **then**
          m_v ++;

From the 33 aircraft, we uniformly at random pick 3 aircraft and artificially model them as light aircraft (light aircraft are hardly present in the traffic mix at Stockholm Arlanda). Let the set of all 33 aircraft (including 3 light aircraft) be $\overline{A}$. We randomly generate the input instances $\mathcal{I}$ for the time window 5:30-5:59AM, during which 17 aircraft arrived in the real-world data, by Algorithm 1. For each selected subset of aircraft A_i and each value of μ, Algorithm 1 creates instances I_i^μ with corresponding arrival times, which we then solve using model $\mathrm{M}_{\mathrm{all}}$.

Algorithm 1 is implemented in Python 3.9.6. We solve the MIP models with Gurobi 13.0.1. All runs were executed on a MacBook Pro M4 (2025).

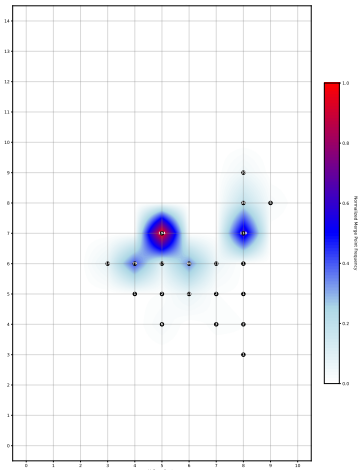


Fig. 2: Heatmap of merge points of all arrival trees computed for the instances $\mathcal{I}$

We ran the algorithm with $|\mathcal{I}| = 300$. The resulting heatmap for used merge points is presented in Figure 2, where each grid node is labeled with the number of times it appears as a merge point in the generated trees. This set of instances resulted in 20 candidate merge points, i.e, for all feasible instances I_i^{μ}, the merge points of the DARs computed with $\mathrm{M}_{\mathrm{all}}$ were a subset of these 20 grid points.

5.2 Dynamic Arrival Routes with Merge-Point Candidates

The implementation of the optimization model is done in Python 3.13.7. We solve the MIP models with Gurobi 12.0.3 using the `gurobipy` interface. All experiments were executed on a MacBook Pro M1 (2020).

In this paper, we present a proof of concept for restricting arrival-tree merge points to a grid subset of candidates, with the focus on the feasibility of computing this set and integrating it into the optimization framework. In real-world applications, the set of aircraft used in the instances in $\mathcal{I}$ should reflect both the traffic mix of the airport and which aircraft types typically appear together in certain time intervals. Hence, we use the same real-world traffic data as in Subsection 5.1, i.e, 33 arrival aircraft in the Stockholm Arlanda TMA during the busiest hour of operation in 2018 (May 16, 05:00–05:59), now with the original arrival times to the entry points. In order to show that different aircraft type with different separation minima (which complicate scheduling) can be handled, we artificially model a random subset of nine aircraft as light aircraft.

We divide the data into two consecutive half-hour periods: 05:00–05:29 and 05:30–05:59. The first period contains 16 arrivals (6 modelled as light) and the second period contains 17 arrivals (3 modelled as light). And we compute the DARs for each of these periods separately (we refer to [5] for approaches of handling consistency between arrival trees for consecutive time periods).

Table 1: Results of the experiments for arrivals between 5:00 AM and 5:29 AM

$\bar{m}$	$\lvert\mathcal{M}\rvert$	μ	objective	tree weight	paths length	average time in TMA (m)	run time (s)
0	165	1	167.376	33.728	182.225	20.6	67
		2	149.026	31.142	162.125	13.5	70
		3	149.026	31.142	162.125	13.5	126
		4	149.026	31.142	162.125	13.5	245
		5	139.567	28.728	151.882	8.8	130
1	20	1	167.576	35.728	182.225	20.6	100
		2	149.026	31.142	162.125	13.5	55
		3	149.026	31.142	162.125	13.5	97
		4	149.026	31.142	162.125	13.5	183
		5	139.567	28.728	151.882	8.8	174
2	16	1	167.676	36.728	182.225	20.6	149
		2	149.026	31.142	162.125	13.5	62
		3	149.026	31.142	162.125	13.5	104
		4	149.026	31.142	162.125	13.5	181
		5	139.567	28.728	151.882	8.8	177
3	13	1	167.676	36.728	182.225	20.6	133
		2	149.026	31.142	162.125	13.5	56
		3	149.026	31.142	162.125	13.5	103
		4	149.026	31.142	162.125	13.5	183
		5	139.567	28.728	151.882	8.8	180
5	12	1	167.676	36.728	182.225	20.6	127
		2	149.026	31.142	162.125	13.5	54
		3	149.026	31.142	162.125	13.5	102
		4	149.026	31.142	162.125	13.5	205
		5	139.567	28.728	151.882	8.8	171
6	11	1	167.676	36.728	182.225	20.6	143
		2	149.026	31.142	162.125	13.5	51
		3	149.026	31.142	162.125	13.5	100
		4	149.026	31.142	162.125	13.5	192
		5	139.567	28.728	151.882	8.8	168
10	10	1	167.676	36.728	182.225	20.6	148
		2	149.026	31.142	162.125	13.5	49
		3	149.026	31.142	162.125	13.5	98
		4	149.026	31.142	162.125	13.5	187
		5	139.567	28.728	151.882	8.8	169
11	9	1	167.676	36.728	182.225	20.6	136
		2	149.026	31.142	162.125	13.5	70
		3	149.026	31.142	162.125	13.5	99
		4	149.026	31.142	162.125	13.5	157
		5	139.567	28.728	151.882	8.8	168
13	8	1	167.676	36.728	182.225	20.6	80
		2	149.026	31.142	162.125	13.5	62
		3	149.026	31.142	162.125	13.5	94
		4	149.026	31.142	162.125	13.5	170
		5	139.567	28.728	151.882	8.8	170
14	7	1	167.676	36.728	182.225	20.6	110
		2	149.026	31.142	162.125	13.5	64
		3	149.026	31.142	162.125	13.5	97
		4	149.026	31.142	162.125	13.5	201
		5	139.567	28.728	151.882	8.8	173
17	6	1	167.676	36.728	182.225	20.6	123
		2	151.446	32.970	164.61	13.5	55
		3	151.446	32.970	164.61	13.5	125
		4	151.446	32.970	164.61	13.5	263
		5	139.567	28.728	151.882	8.8	169

Table 1: Results of the experiments for arrivals between 5:00 AM and 5:29 AM

$\bar{m}$	$\lvert\mathcal{M}\rvert$	μ	objective	tree weight	paths length	average time in TMA (m)	run time (s)
36	5	1	167.676	36.728	182.225	20.6	108
		2	151.446	32.970	164.61	13.5	56
		3	151.446	32.970	164.61	13.5	126
		4	151.446	32.970	164.61	13.5	256
		5	139.567	28.728	151.882	8.8	168
66	4	1	167.676	36.728	182.225	20.6	136
		2	151.446	32.970	164.61	13.5	58
		3	151.446	32.970	164.61	13.5	119
		4	151.446	32.970	164.61	13.5	360
		5	139.567	28.728	151.882	8.8	179
79	3	1	171.486	37.556	186.367	20.6	300
		2	151.446	32.970	164.61	13.5	103
		3	151.446	32.970	164.61	13.5	120
		4	151.446	32.970	164.61	13.5	214
		5	139.567	28.728	151.882	8.8	182

Table 2: Results of the experiments for arrivals between 5:30 AM and 5:59 AM

$\bar{m}$	$\lvert\mathcal{M}\rvert$	μ	objective	tree weight	paths length	average time in TMA (m)	run time (s)
0	165	≤ 3	—				
		4	173.886	34.556	189.367	11.9	372
		5	154.867	28.728	168.882	9.3	232
1	20	4	173.886	34.556	189.367	11.9	465
		5	154.867	28.728	168.882	9.3	230
2	16	4	173.886	34.556	189.367	11.9	389
		5	154.867	28.728	168.882	9.3	264
3	13	4	173.886	34.556	189.367	11.9	359
		5	154.867	28.728	168.882	9.3	236
5	12	4	173.886	34.556	189.367	11.9	412
		5	154.867	28.728	168.882	9.3	232
6	11	4	173.886	34.556	189.367	11.9	477
		5	154.867	28.728	168.882	9.3	243
10	10	4	173.886	34.556	189.367	11.9	414
		5	154.867	28.728	168.882	9.3	232
11	9	4	173.886	34.556	189.367	11.9	452
		5	154.867	28.728	168.882	9.3	231
13	8	4	173.886	34.556	189.367	11.9	412
		5	154.867	28.728	168.882	9.3	230
14	7	4	173.886	34.556	189.367	11.9	367
		5	154.867	28.728	168.882	9.3	229
17	6	4	173.986	35.556	189.367	11.9	404
		5	154.867	28.728	168.882	9.3	228
36	5	4	173.986	35.556	189.367	11.9	425
		5	154.867	28.728	168.882	9.3	231
66	4	4	173.986	35.556	189.367	11.9	389
		5	154.867	28.728	168.882	9.3	250
79	3	4	173.986	35.556	189.367	11.9	371
		5	154.867	28.728	168.882	9.3	225

We construct the candidate set $\mathcal{M}$ for each run using the heatmap from Subsection 5.1. Since only 20 grid nodes appear as merge points for instances in $\mathcal{I}$, the candidate set does not change for every threshold value $\bar{m}$. We therefore consider only values for which $\mathcal{M}$ changes. For each such $\bar{m}$, we solve model $\mathrm{M}_{\mathrm{cand}}$ with different time-window offsets μ. We also include the case $\bar{m} = 0$, corresponding to model $\mathrm{M}_{\mathrm{all}}$, i.e., the unrestricted case, which yields a globally optimal solution for the considered data and serves as a reference for evaluating the impact of restricting the merge-point set.

Tables 1 and 2 report the computational results for the two time periods. Each table shows the threshold $\bar{m}$, the number of candidate points $|\mathcal{M}|$, the time-window offset μ, the total objective value, the tree weight, the total paths lengths, the average time (in minutes) each aircraft spends in the TMA, and the run time for solving $\mathrm{M}_{\mathrm{cand}}$ (in seconds). Since no feasible solutions exist for the second half-hour with $\mu \leq 3$, we do not include these cases in Table 2. We also provide the resulting optimal arrival trees for some of our results in Figures 3–4.

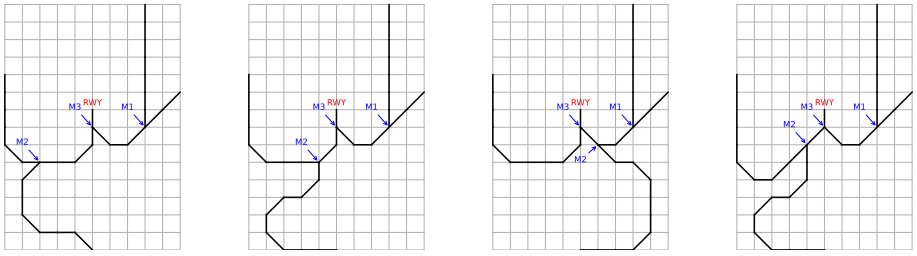
The parameter $\bar{m}$ controls the maximum size of the candidate set used when constructing merge points and, therefore, directly influences the feasible merge structures. When $\bar{m} = 0$, no candidate set is imposed and every node in the grid graph may serve as a merge point. As $\bar{m}$ increases, the set of admissible merge points becomes progressively restricted, which alters both the topology of the resulting merge tree and the flexibility in selecting aircraft paths.

Objective values may coincide for different values of $\bar{m}$ but this does not imply that the resulting trees are identical. For a given μ, a tree that is optimal for $\bar{m} = m^*$ is feasible also for all $\bar{m} < m^*$, and if the objective value coincides, it is also an optimal tree for that $\bar{m} < m^*$. However, the opposite is not true, and we may not always inherit trees from smaller values of $\bar{m}$ because some used merge-point candidates may no longer be present. Moreover, for relatively small $\bar{m}$, we may achieve the global optimal objective value. However, when $\bar{m}$ becomes large, deviations appear for some values of μ, leading to slightly larger objective values. This indicates that the restriction on merge points eventually affects the tree structure and overall solution quality. Hence, there is a tradeoff between the complexity for ATCOs (in terms of the number of grid points selected as merge-point candidates) and the quality of aircraft trajectories in the TMA.

The parameter μ introduces a second trade-off: increasing μ improves the objective value and reduces tree weight and average time in the TMA by allowing more flexible entry times, but at a cost for aircraft operators, as larger offsets may require speed adjustments or path stretching.

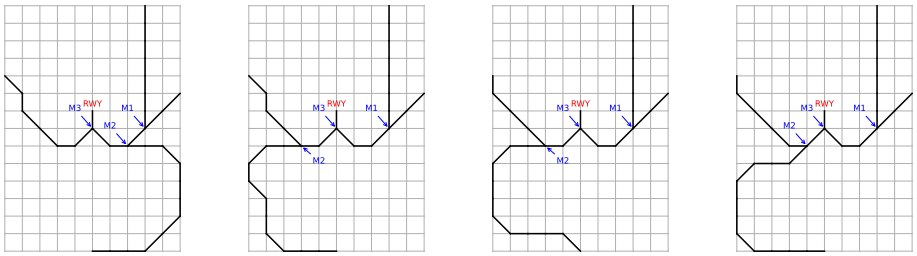
6 Conclusions

We presented an approach for reducing complexity of DARs for ATCOs by restricting merge points of the entry-point–runway paths to a set of candidates, following recommendations from operational ATCOs. We give a proof of concept of a data-driven approach to determine candidate merge points and of integrating this candidate set into our framework to compute DARs.



(a) (b) (c) (d)

Fig. 3: Optimal arrival trees for 5:00–5:29 with $\mu = 1$, (a) all points as candidates, (b)-(d) with merge point candidates obtained by $\bar{m} = 1, 11$, and 79, respectively



(a) (b) (c) (d)

Fig. 4: Optimal arrival trees for 5:30–5:59 with $\mu = 4$, (a) all points as candidates, (b)-(d) with merge point candidates obtained by $\bar{m} = 1, 11$, and 79, respectively

With our approach, we aim to balance the flexibility of DARs, which allows us to schedule and automatically separate all aircraft following environmentally friendly, fuel-optimized descent profiles even in high-traffic scenarios, and the complexity of the resulting arrival-route structure for ATCOs. Since merge points require the most attention of ATCOs, restricting them to a few spots has the potential to reduce complexity. Moreover, DARs with merge-point candidates still allow the arrival routes to occupy varying parts of the TMA during different time periods, which can accommodate a wide variety of traffic scenarios.

With experiments for Stockholm Arlanda airport, we verify that we can obtain feasible solutions for different cardinalities (with $|\mathcal{M}| \geq 3$) of the merge-point candidate set. We observe the expected tradeoff between complexity and the quality of aircraft trajectories: the more well-chosen merge-point candidates we have, the better the objective function and performance measures like the average time in TMA, but also the higher the complexity. Consequently, identifying an appropriate balance between the size of the candidate set, DARs performance and operational complexity, as well as between sequencing flexibility and schedule stability, remains a key direction for future research.

References

1. EASA: European aviaton environmental report 2025. https://www.eurocontrol.int/sites/default/files/2025-01/eurocontrol-easa-eaer-2025.pdf
2. Eurocontrol: Point merge implementation: A quick guide, simplifying and enhancing arrival operations with closed loop sequencing
3. Eurocontrol: Point merge, improving and harmonising arrival operations. https://www.eurocontrol.int/concept/point-merge
4. Hajizadeh, R., Hardell, H., Polishchuk, T., Rönnberg, E., Schmidt, C.: Integrating atmospheric conditions into the dynamic-arrival-routes framework. In: SIDs (2025)
5. Hajizadeh, R., Polishchuk, T., Rönnberg, E., Schmidt, C.: A Dantzig-Wolfe reformulation for automated aircraft arrival routing and scheduling. unpublished (2025), https://arxiv.org/abs/2509.13065
6. Hardell, H., Otero, E., Polishchuk, T., Smetanova, L.: Optimizing air traffic management through point merge procedures: Minimizing delays and environmental impact in arrival operations. Journal of Air Transport Management **123** (2025)
7. IATA: Global outlook for air transport: Highly resilient, less robust (2023)
8. ICAO: Resolution a41-21: Consolidated statement of continuing icao policies and practices related to environmental protection — climate change. https://www.icao.int/sites/default/files/sp-files/environmental-protection/Documents/Assembly/Resolution_A41-21_Climate_change.pdf
9. Liu, W., Delahaye, D., Cetek, F.A., Zhao, Q., Notry, P.: Comparison of performance between PMS and trombone arrival route topologies in terminal maneuvering area. Journal of Air Transport Management **115**, 102532 (2024)
10. OpenSky Network. https://opensky-network.org (Jul 2019), acc. December 2, 2018
11. Prete, J., Krozel, J., Mitchell, J., Kim, J., Zou, J.: Flexible, Performance-Based Route Planning for Super-Dense Operations. In: AIAA Guidance, Navigation and Control Conference and Exhibit. American Institute of Aeronautics and Astronautics (Aug 2008)
12. Sáez, R., Prats, X., Polishchuk, T., Polishchuk, V.: Traffic Synchronization in Terminal Airspace to Enable Continuous Descent Operations in Trombone Sequencing and Merging Procedures: An Implementation Study for Frankfurt Airport. Transportation Research Part C: Emerging Technologies **121** (2020)
13. SKYbrary: Icao wake turbulence categories. https://skybrary.aero/articles/icao-wake-turbulence-category
14. SKYbrary: Mitigation of wake turbulence hazard, wake turbulence separation minima. https://skybrary.aero/articles/mitigation-wake-turbulence-hazard, acc. Sept 22, 2025
15. Sprong, K., Haltli, B., DeArmon, J., Bradley, S.: Improving flight efficiency through terminal area rnav. In: 6th USA/Europe ATM Seminar (2005)
16. Sáez, R., Polishchuk, T., Schmidt, C., Hardell, H., Smetanová, L., Polishchuk, V., Prats, X.: Automated sequencing and merging with dynamic aircraft arrival routes and speed management for continuous descent operations. Transportation Research Part C: Emerging Technologies **132**, 103402 (2021)

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Mathematical Techniques for Evaluating University Timetabling Policies

Matthew Davison[1,2], Ahmed Kheiri[3], and Konstantinos G. Zografos[4,5]

[1] Department of Mathematics, Aarhus University, Aarhus, Denmark
md@math.au.dk
[2] STOR-i Centre for Doctoral Training, Lancaster University, Lancaster, UK
[3] Alliance Manchester Business School, University of Manchester, Manchester, UK
ahmed.kheiri@manchester.ac.uk
[4] Edinburgh Business School, Heriot-Watt University, Edinburgh, UK
k.zografos@hw.ac.uk
[5] Distinguished Professor Emeritus, Department of Management Science, Lancaster University, Lancaster, UK

Abstract. University timetabling is a critical operational challenge, with policy decisions directly impacting staff, students, and resources. This paper presents a new benchmark instance derived from real-world data (*lancs-yr23*). Using this instance alongside an existing timetabling model and solution methods, we assess four policy questions: curriculum restructuring, enforcement of core teaching hours, relaxation of room capacity, and hybrid teaching under reduced physical capacity. This is done by modifying the model structure or data to represent each problem and solving the altered model. Our results show that students remain highly interconnected regardless of curriculum changes, that intuitive scheduling improvements can have counterproductive effects, and that modest capacity relaxation yields meaningful gains. *lancs-yr23* and the methodology provide a reusable framework for similar problems.

Keywords: Policy evaluation · Decision-making · Timetabling.

1 Introduction

The UK higher education (HE) sector generated £52 billion in total income in 2023/24 [22]. Tuition fees and education contracts account for over 52% of total income, dominating other income sources. Universities aim to attract as many students as possible to maximise this income.

A significant reason students choose a university is the quality of its teaching [2]. High-quality teaching, characterised by effective instruction and classroom management [6], requires knowledgeable staff and modern resources, both of which are costly. Despite the increasing income of the UK HE sector, Figure 1 shows that expenditure nearly matches and occasionally exceeds income. Data for 2023/24 shows that £18 billion went towards staff wages and £20 billion was spent on operating expenses [21]. This strongly motivates the need for careful staff and resource management.

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

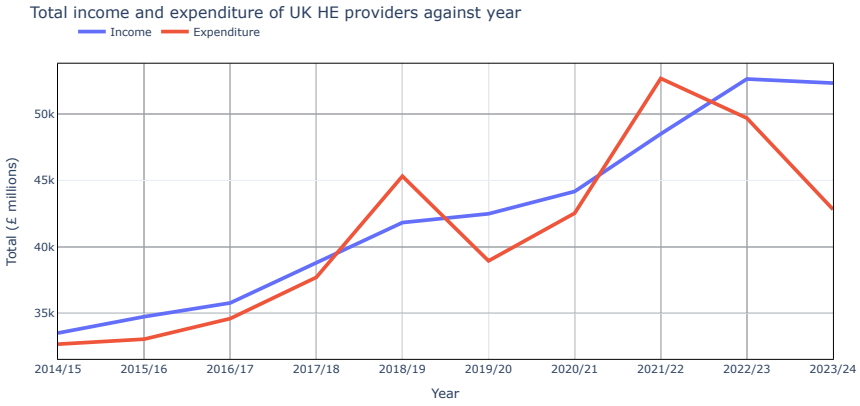


Fig. 1: The total income [22] and total expenditure [21] for higher education providers in the United Kingdom.

A central operational element at any university is the teaching timetable. This timetable dictates what staff, students and resources are doing at any given time. The construction of this timetable is limited by university policy. Consequently, modifying any university policy could have either a positive or detrimental impact on operational costs. This motivates the following research questions:

- Can existing timetables be used to identify specific policies problematic to university operation?
- Can changes in particular policies be modelled so that these changes can be incorporated into a timetabling model?
- Can the resulting timetables produced by this timetabling model be used to assess the overall impact of various policy changes?

To address these questions, this paper aims to: (i) reshape real-world timetabling data into a standard format, forming a novel benchmark instance, (ii) use this instance and knowledge from the university timetabling department to identify policies causing operational bottlenecks, and (iii) model and test changes to these policies, using change in timetable quality as a proxy for impact. Sections 2–5 cover related work, instance construction, policy evaluation, and conclusions, respectively.

2 Related work

For overviews of operational research in education, university course timetabling problem (UCTTP) methods, and benchmarking datasets, see [16], [5], and [4], respectively. Historically, timetabling was conducted manually from the prior

schedule; [8] and [3] promote automatic methods to address the resulting inefficiencies. Building on this, researchers moved to developing models and methods that not only build schedules but act as tools for decision support.

The most direct approach is to build a new model tailored to the decision at hand. The model in [15] enabled the US Air Force Academy to switch from an alternating-day to a repeated-week pattern, yielding cost savings and class consolidations. [1] develop a mixed integer program (MIP) to assess a semester calendar change at MIT under COVID-induced resource scarcity.

Sensitivity analysis by modifying input parameters of these models can also produce insights. [23] use an integer programming timetabling model with varied input data to assess whether a schedule can be found if certain rooms are unavailable. Their case study helped identify teaching spaces the school could relinquish, albeit with some impact on timetable quality.

If a model already exists, some researchers modify or introduce constraints or objectives to represent new features. [7] optimise soft constraints relating to "compactness" (the arrangement of free time between activities), finding that their heuristic outperforms manual scheduling. [24] incorporate student flow modelling into an MIP for timetabling, significantly reducing corridor congestion. [18] modify the objective function of the timetabling problem described in [13] in various ways to incorporate "fairness". They then use this model to investigate the trade-off between timetable efficiency and fairness. [14] also pursue fairness in course timetabling via both modified objectives and additional constraints.

When several models share variables, these can be combined into a single monolithic model. [17] combine MIP models to explore trade-offs between different timetabling objectives.

This review highlights several ways practitioners can assess policy and other timetabling changes:

- Create a new model to replace outdated models.
- Sensitivity analysis by modifying input parameters.
- Modifying model aspects such as the objective function or constraints.
- Combine multiple models into a single model.

2.1 Contributions

The key gaps are the following: authors typically apply a single technique, give little guidance on which technique suits a given strategic problem, and seldom release case study data. This paper addresses all three, with the following contributions:

- A new benchmark instance for a large campus-based university in the United Kingdom and a thorough description of its creation.
- A demonstration of how to analyse the instance to identify bottlenecks caused by university policy or infrastructure.
- A study using the new instance in which we select and combine techniques to assess a range of policy changes, offering guidance on which technique suits which problem.

3 Instance construction and validation

The 2023/24 timetable at Lancaster University (LU) was provided by the LU timetabling team and exported from Scientia Syllabus Plus (SS+) as three files describing room, student and activity attributes. Some of this data was filtered in consultation with the timetabling team. Converting these exports into the format of the International Timetabling Competition 2019 (ITC-2019) [19] follows a largely routine extraction. We therefore restrict this section to the non-obvious decisions that depart from the ITC-2019 specification or that materially shaped the resulting benchmark.

3.1 Real-world data challenges

Two issues occur in real timetabling exports. Firstly, the timetable contains "tricks" used to force activities into the system which were removed on the advice of the timetabling team. For example, some field trips were booked as full-day campus events. Secondly, although no room may host two activities simultaneously in [11], the raw timetable violates this for a small number of spaces. Perfectly overlapping activities are treated as identical and merged, while partial overlaps are trimmed or split into parts, affecting under 0.02% of activities.

Room capacity was in the export, but availability was not, so rooms are assumed to be always available. Travel time within a building is taken as zero, and travel between buildings is computed from OpenStreetMap (OSM) data [20]. This is done by identifying the network node nearest to the centroid of each building, and the shortest-path time between nodes gives the building-to-building travel time (Figure 2). Spaces flagged with the "Lecture Capture or Stream" suitability are tagged as "hybrid capable", a property used in Section 4 and a room property not in the ITC-2019 specification.



Fig. 2: Map of the Lancaster campus at Lancaster University in the United Kingdom [20] with an example route between two buildings.

3.2 Adapting the ITC-2019 format to the model of [11]

The model of [11] shares many features with the ITC-2019 format, so most fields carry over directly; however, a few points need adaptation. The most substantial concerns module structure, which is inferred from the activities each student attends. Every distinct combination of activities forms a *module configuration*, with each activity a subpart. Because the raw data reuses activity identifiers across students, duplicates are relabelled to keep names unique and tied together by a "SameClass" constraint, equivalent to imposing the ITC-2019 "SameStart", "SameTime", "SameDays", "SameWeeks" and "SameRoom" constraints on the pair at once.

Time in the ITC-2019 is handled by describing each session with a start, a duration, and binary day and week patterns ("1001000" denoting Monday and Thursday); the set of times an activity may take forms its timeset. As a session carries a single start, duration and pattern, any activity whose sessions vary in these is split into separate sessions that a student must attend together. Rooms are assigned by giving each activity its largest suitable space or none (treated as online). Instructors shared across activities induce constraints that keep those activities from overlapping in time or sitting too far apart to travel between.

Finally, to give the optimiser room to manoeuvre in the experiments of Section 4, each activity is assigned candidate rooms (same zone, sufficient capacity, meeting all suitability requirements) and candidate timesets, generated by shifting its start, permuting its days, or permuting its active weeks while preserving weekly contact hours.

3.3 The *lancs-yr23* instance

The resulting instance, *lancs-yr23*, covers 65 weeks. On average, there are 7.39 modules per student and 5.74 activities per staff member. Other summary statistics are given in Table 1, and the instance is publicly available on the Lancaster University Research Directory [9].

Table 1: Summary metrics of the instance created using university data.

Metric	Activities	Modules	Spaces	Students	Timesets	Staff
Value	60,728	1,566	640	13,991	35,539	410

The LU 2023/24 timetable is a feasible solution to *lancs-yr23*, encoded in the instance as assignments with zero penalty. This is used as the initial solution throughout: both as the starting point required by the matheuristic of [10] and as the baseline against which policy changes are measured.

4 Policy evaluation

Using the techniques identified in Section 2, we assess changes to university timetabling policies by using *lancs-yr23* with some adaptation of the UCTTP model of [11] and solving or analysing the resulting model using the methodology of [10]. Each experiment modifies the model to capture the policy under study. We describe each modification in words and refer the reader to those works for the exact formulations.

Three objectives appear throughout this paper. z_1 is the number of elective module requests met (maximised); z_2 is the number of deviations from students' preferred teaching mode (minimised); and z_3 is the number of student scheduling issues (minimised). An elective module request is a module a student would like to take; teaching mode is how an activity is delivered, and a scheduling issue is a pair of a student's activities that overlap in time or are placed too far apart to allow travel between them.

The original timetable supplies a baseline, and unless stated otherwise, each experiment holds z_1, z_2 and z_3 to be no worse than their baseline values, isolating the effect of the policy change. After presenting each experiment, we review our findings and discuss the implications for strategic planning.

4.1 Curricula changes

Curriculum changes significantly affect timetabling, since university students often select which modules they take within their course. [10] proposes a measure for how "related" any two students are based on their module selection, together with a method for partitioning students on this measure. Using that methodology, we ask how related students are at present, how that relatedness changes under a modified curriculum, and the implications for the timetabling process.

In the raw LU data, the specific programme is given by the first four characters of the module, indicating the school or theme. For example, "MSCI222" is a module under "Management Science". The maximum number of schools/themes per student is eight. Let $n \in \{1, \ldots, 8\}$. For each n, an instance is created keeping only each student's top n schools/themes: if a student requests $\{\text{MSCI222}, \text{MSCI223}, \text{MATH240}\}$ and $n = 1$, the requests reduce to $\{\text{MSCI222}, \text{MSCI223}\}$, as "MSCI" occurs twice and "MATH" once. Students are then partitioned into module-independent blocks. Figure 3 shows how the number of blocks changes with n. The partitions created for $n \in \{3, \ldots, 8\}$ were all identical.

Table 2 shows that the largest partition block contains 91% and 93% of students for $n = 2$ and $n = 3$, respectively, indicating that any two students are likely connected by some chain of modules and that a clean partition is therefore difficult to achieve. Only when $n = 1$ does the method partition the students effectively, with the largest block in this case holding 6% of students.

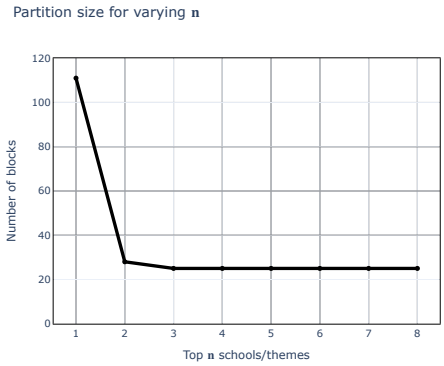


Fig. 3: Line plot illustrating the number of blocks in the partition when the top n schools/themes of each student are kept.

Table 2: Summary statistics for the partitions found.

Value of n	Block count	Largest block size	Smallest block size	Average block size
1	111	858	1	126
2	28	12,758	3	500
3	25	12,969	3	560

This experiment demonstrates that even radical curriculum restructuring leaves students highly interconnected through shared modules, so restricting interdisciplinary study would not meaningfully reduce timetabling complexity.

4.2 Teaching hours

At many universities, there are core teaching hours. At LU, these are 09:00 to 18:00 Monday to Friday, except Wednesday (09:00 to 13:00). The timetabling policy permits credit-bearing teaching outside these hours, but out-of-hours activities are disruptive: the Wednesday afternoon gap is reserved for extra-curricular activities, and early or late sessions disadvantage commuting students. In the baseline timetable, 3,814 activities (6% of the total) fall outside core hours, yet these affect 9,740 unique students (70%). Motivated by these figures, we ask whether activities can be moved within teaching hours and what the effect on infrastructure usage is.

One approach is to attach an integer penalty to each timeset equal to the number of meetings it places outside core hours. For a class held after 18:00 on a Monday and a Tuesday for three weeks, the penalty is

$$1\,(\text{event}) \times 2\,(\text{days}) \times 3\,(\text{weeks}) = 6. \tag{1}$$

We then minimise the sum of these penalties by applying the matheuristic of [10], which reduced the number of out-of-hours events from 3,814 to 3,751.

However, the new solution requires 35 more spaces (a 13% increase) and affects slightly more students (9,776 versus 9,740). This is a clear case of an intuitive improvement producing counterproductive results in practice.

4.3 Capacity relaxation

A key restriction at many universities is the availability of suitably sized teaching spaces. In the raw LU data, more than 400 activities had a planned or actual size exceeding the capacity of any suitable space. Figure 4 suggests why: many modules compete for the limited number of high-capacity venues on campus.

Room capacity is sometimes treated as a soft constraint in the literature [12], because attendance peaks at the start of a semester, drops, and then stabilises. The hard capacity constraints in this model create a tension between long-term efficiency and immediate student satisfaction. We therefore ask how many activities are over-subscribed, how necessary a one-to-one seating policy is, and whether a drop-off in attendance can be designed into the timetable.

Relaxing capacity first requires giving activities access to larger rooms, so we expand each activity's candidate locations to include rooms with enough capacity for its planned or actual attendance, whichever is smaller. Comparing each affected activity's attendance against its largest candidate room yields a set of *capacity deviations*. These distinct values, sorted into ascending order and prefixed with zero, form the relaxation levels $\mathcal{C}^{\text{dev}}$.

In [11], each activity's in-person attendance is bounded by the capacity of its assigned room. We relax this by granting each activity c a non-negative integer slack Δ_c above its room capacity, subject to a shared ceiling $\Delta_c \leq \Delta$. Setting $\Delta = 0$ recovers the original hard constraint.

For each level $\delta \in \mathcal{C}^{\text{dev}}$, we fix $\Delta = \delta$, hold z_1, z_2 and z_3 to be no worse than baseline, and minimise z_2. Table 3 reports that relaxing capacity increases the number of mode preferences met. As Δ grows, more activities exceed their original capacity while the average relaxation used becomes a smaller fraction of the maximum permitted.

Table 3: Level of relaxation (Δ_c), total number of deviations from a student's preferred mode (z_2), number of activities with relaxed capacities, maximum relaxation and average relaxation.

	Δ_c			
	0	1	13	28
z_2	83,230	81,714	80,680	79,816
Relaxed capacities	0	303	501	655
Maximum relaxation	0	1	13	28
Average relaxation	0	1	5.54	8.04

Figure 5 shows how far each relaxed activity's attendance exceeds its room capacity. As the maximum permitted relaxation Δ increases, activities use a progressively smaller share of it, so most require only modest violations. A small Δ therefore secures much of the gain in mode preferences. Unlike a fully soft capacity constraint, this approach also bounds the worst-case overrun: a practitioner can read an acceptable Δ from such a histogram and continue optimising at that level.

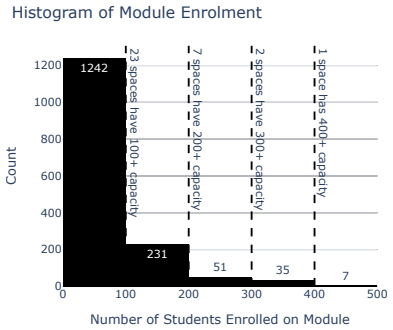


Fig. 4: Histogram showing the module enrolment figures. Information on physical space capacity is also given to highlight the issue of limited resources.

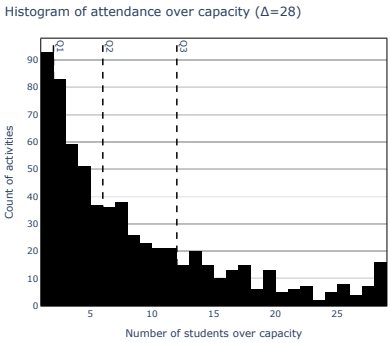


Fig. 5: Histogram showing the number of students over capacity for each relaxed activity. Quantiles are marked using dashed lines.

4.4 Hybrid teaching

Where the previous section addressed declining attendance, here we consider a drop in available physical capacity. During the 2020 COVID pandemic, university modules moved online to enable social distancing; on returning, some universities offered "hybrid teaching", in which part of a class attends in person and part online. Spreading students out for safe in-person attendance essentially reduced physical capacity. In some cases, this was a fourfold reduction [1]. Figure 6 shows that the baseline timetable becomes infeasible under such a reduction.

Some universities, including LU, have spaces that can record or broadcast lectures, though these are a small share of teaching spaces (19% at LU). We ask whether hybrid teaching resolves the capacity issue, whether the spaces to upgrade can be identified, and whether the number of upgrades can be minimised.

Solution under scarcity: Reducing every physical space's capacity by 75% makes the current solution infeasible, so everything is first moved online, and we then aim to return as many students as possible to in-person teaching. Concretely, we minimise the deviations from preferred mode (z_2) while holding the

number of module requests met and the number of student conflicts to be no worse than their baseline values. Starting from an all-online solution avoids fixing the in-person/online split in advance, where a poor split could introduce conflicts through large travel times.

After some optimisation, only 50 activities returned to in-person teaching and 5 to hybrid mode. This limited gain could be a result of restrictive student timetables and the overall reduced capacity. As we are using a large-neighbourhood search, we are restricted to a limited number of iterations. Figure 7 shows that the objective does not plateau in the way a matheuristic near optimality usually does, suggesting that further iterations could restore more activities.

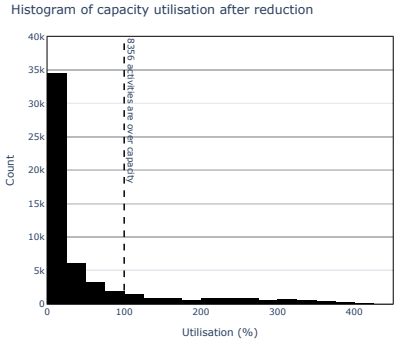


Fig. 6: Histogram showing theoretical utilisation of a space after reducing physical capacity by 75%.

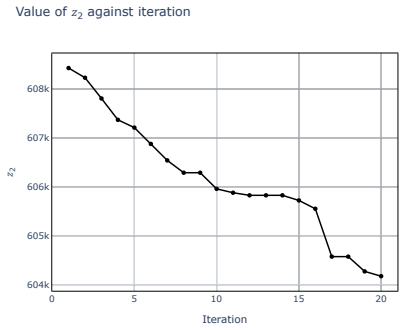


Fig. 7: Plot of the objective value against iteration in the case of restricted resources.

Solution allowing conversions: Not every space can host hybrid teaching, and conversions are costly, so we seek the fewest conversions necessary. We add a binary variable ∇_r marking each non-hybrid room r as converted (those in $R \backslash R^h$, where R^h is the set of hybrid-capable spaces), and relax the hybrid-compatibility constraint of [11] so a converted room may be used for hybrid teaching. We first allow conversions freely while improving z_2 and once this stalls, we fix z_2 and minimise the number of conversions,

$$z_5 = \sum_{r \in R \backslash R^h} \nabla_r. \tag{2}$$

Minimising z_5 identifies the essential conversions. Further optimisation improved z_2, returning 145 activities to in-person teaching (195 in total) and 26 to hybrid mode (31 in total). Post-optimisation analysis found that seven rooms had been converted for hybrid teaching, and we were unable to reduce this number by minimising z_5. Figure 8 shows the impact of each conversion on student and activity counts, identifying priority areas for hybrid enhancement. Spaces

113 and *253* each serve 11 activities and affect over 90 students, so they should be prioritised over the others.

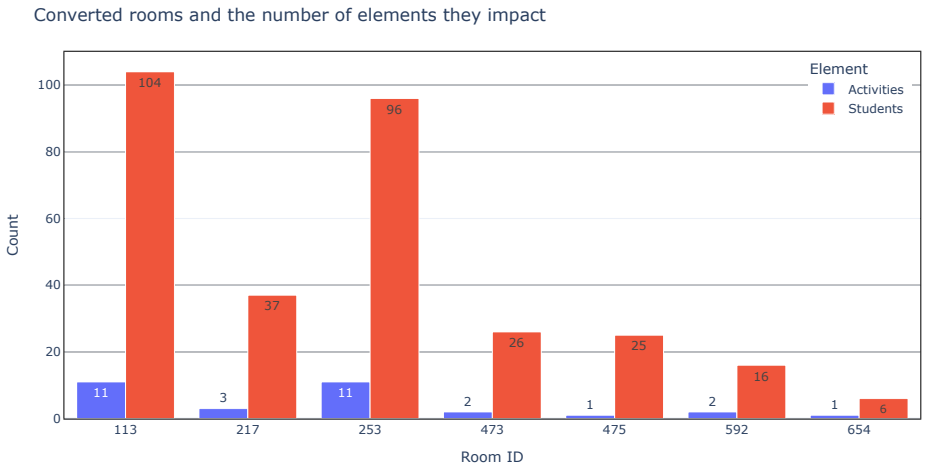


Fig. 8: Converted rooms and the number of activities using them, and the number of students this impacts.

5 Conclusion

This paper builds a new ITC-2019 benchmark instance from Lancaster University data, *lancs-yr23*, and uses it to assess four policy questions, drawing together the technique families of Section 2 in one case study. Three of the four build on the UCTTP model of [11]: the teaching-hours and capacity problems extend it with new objectives or constraints, and the hybrid-teaching problem combines several techniques at once. The curriculum problem instead modifies the input data and applies the partitioning scheme of [10], a form of sensitivity analysis. Three takeaways stand out:

- Building the benchmark was more than a format conversion: real timetables contain workarounds and overlapping bookings that must be cleaned up without distorting the problem.
- Intuitive fixes can make things worse. Moving teaching into core hours cut out-of-hours events but required more rooms and affected more students.
- The harder questions needed several techniques together rather than one. The hybrid-teaching study combined a model extension with a new objective to pinpoint the few rooms worth converting.

Future work could turn this into a catalogue pairing policy questions with the techniques best suited to them, helping practitioners choose an approach quickly. The *lancs-yr23* instance is publicly available to support this [9].

Acknowledgements We gratefully acknowledge the support of the EPSRC-funded EP/S022252/1 STOR-i Centre for Doctoral Training. We also thank the timetabling team at Lancaster University for providing us with timetable data and offering insight regarding timetabling challenges at LU.

References

1. Barnhart, C., Bertsimas, D., Delarue, A., Yan, J.: Course scheduling under sudden scarcity: Applications to pandemic planning. Manufacturing & Service Operations Management **24**(2), 727–745 (2022). https://doi.org/10.1287/msom.2021.0996, https://doi.org/10.1287/msom.2021.0996
2. Bhardwa, S.: Why do students go to university and how do they choose which one? Times Higher Education (Jun 2017), https://www.timeshighereducation.com/student/news/why-do-students-go-university-and-how-do-they-choose-which-one, accessed: 2025-06-06
3. Carter, M.W.: A comprehensive course timetabling and student scheduling system at the University of Waterloo. In: Burke, E., Erben, W. (eds.) Practice and Theory of Automated Timetabling III. pp. 64–82. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
4. Ceschia, S., Di Gaspero, L., Schaerf, A.: Educational timetabling: Problems, benchmarks, and state-of-the-art results. European Journal of Operational Research **308**(1), 1–18 (2023). https://doi.org/10.1016/j.ejor.2022.07.011, https://www.sciencedirect.com/science/article/pii/S0377221722005641
5. Chen, M.C., Sze, S.N., Goh, S.L., Sabar, N.R., Kendall, G.: A survey of university course timetabling problem: Perspectives, trends and opportunities. IEEE Access **9**, 106515–106529 (2021). https://doi.org/10.1109/ACCESS.2021.3100613
6. Coe, R., Aloisi, C., Higgins, S., Major, L.E.: What makes great teaching? review of the underpinning research. Tech. rep., Durham Research Methods Centre (DRMC) (2014), https://durham-repository.worktribe.com/output/1607293, full text deposited in DRO
7. Dammak, A., Elloumi, A., Kamoun, H., Ferland, J.: Course timetabling at a Tunisian university: A case study. Journal of Systems Science and Systems Engineering **17**, 334–352 (Sep 2008). https://doi.org/10.1007/s11518-008-5079-5
8. Daskalaki, S., Birbas, T., Housos, E.: An integer programming formulation for a case study in university timetabling. European Journal of Operational Research **153**(1), 117–135 (2004). https://doi.org/10.1016/S0377-2217(03)00103-6, https://www.sciencedirect.com/science/article/pii/S0377221703001036
9. Davison, M.: International Timetabling Competition 2019 approximation of Lancaster University 2023/2024 term. Dataset, file `lancs-yr23.xml` (Nov 2025). https://doi.org/10.17635/lancaster/researchdata/279, https://doi.org/10.17635/lancaster/researchdata/279, created 18 November 2025
10. Davison, M.: Modelling and Solving Multi-objective University Course Timetabling Problems. Ph.D. thesis, Lancaster University (2025). https://doi.org/10.17635/lancaster/thesis/3040
11. Davison, M., Kheiri, A., Zografos, K.G.: Modelling and solving the university course timetabling problem with hybrid teaching considerations. Journal of Scheduling **28**(2), 195–215 (Apr 2025). https://doi.org/10.1007/s10951-024-00817-w, https://doi.org/10.1007/s10951-024-00817-w

12. De Causmaecker, P., Demeester, P., Vanden Berghe, G.: A decomposed metaheuristic approach for a real-world university timetabling problem. European Journal of Operational Research **195**(1), 307–318 (May 2009). https://doi.org/10.1016/j.ejor.2008.01.043, https://www.sciencedirect.com/science/article/pii/S0377221708001665
13. Di Gaspero, L., McCollum, B., Schaerf, A.: The second International Timetabling Competition (ITC-2007): Curriculum-based course timetabling (Track 3). Tech. rep., University of Udine (2007)
14. Gogos, C., Dimitsas, A., Valouxis, C., Alefragis, P.: Modeling a balanced commute educational timetabling problem in the context of teaching integer programming. In: 2022 7th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM). pp. 1–5 (2022). https://doi.org/10.1109/SEEDA-CECNSM57760.2022.9932912
15. Gonzalez, G., Richards, C., Newman, A.: Optimal course scheduling for United States Air Force Academy cadets. Interfaces **48**(3), 217–234 (2018). https://doi.org/10.1287/inte.2017.0935
16. Johnes, J.: Operational Research in education. European Journal of Operational Research **243**(3), 683–696 (2015). https://doi.org/10.1016/j.ejor.2014.10.043, https://www.sciencedirect.com/science/article/pii/S0377221714008650
17. Lindahl, M., Mason, A.J., Stidsen, T., Sørensen, M.: A strategic view of university timetabling. European Journal of Operational Research **266**(1), 35–45 (2018). https://doi.org/10.1016/j.ejor.2017.09.022, https://www.sciencedirect.com/science/article/pii/S0377221717308433
18. Mühlenthaler, M., Wanka, R.: Fairness in academic course timetabling. Annals of Operations Research **239**(1), 171–188 (Apr 2016). https://doi.org/10.1007/s10479-014-1553-2, https://doi.org/10.1007/s10479-014-1553-2
19. Müller, T., Rudová, H., Müllerová, Z.: Real-world university course timetabling at the International Timetabling Competition 2019. Journal of Scheduling **28**(2), 247–267 (Apr 2025). https://doi.org/10.1007/s10951-023-00801-w, https://doi.org/10.1007/s10951-023-00801-w
20. OpenStreetMap contributors: Way: Lancaster University (23347159). https://www.openstreetmap.org/way/23347159 (2025), accessed: 2025-06-06
21. Perrott, L.: What is the expenditure of HE providers? Higher Education Statistics Agency (HESA) (May 2025), https://www.hesa.ac.uk/data-and-analysis/finances/expenditure
22. Perrott, L.: What is the income of HE providers? Higher Education Statistics Agency (HESA) (May 2025), https://www.hesa.ac.uk/data-and-analysis/finances/income
23. Schimmelpfeng, K., Helber, S.: Application of a real-world university-course timetabling model solved by integer programming. OR Spectrum **29**(4), 783–803 (Oct 2007). https://doi.org/10.1007/s00291-006-0074-z, https://doi.org/10.1007/s00291-006-0074-z
24. Vermuyten, H., Lemmens, S., Marques, I., Beliën, J.: Developing compact course timetables with optimized student flows. European Journal of Operational Research **251**(2), 651–661 (2016). https://doi.org/10.1016/j.ejor.2015.11.028, https://www.sciencedirect.com/science/article/pii/S0377221715010772

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Reentrant Permutation Flow Shop Problems with the Total Completion Time Objective

Itamar Segal[1], Tal Grinshpoun[1][0000−0002−4106−3169], Hagai Ilani[2][0000−0003−3548−1572], and Elad Shufan[2][0000−0001−7960−5798]

[1] Ariel University, Ariel, Israel
itamar.segal@msmail.ariel.ac.il, talgr@ariel.ac.il
[2] Shamoon College of Engineering, Ashdod, Israel
hagai@sce.ac.il, elads@sce.ac.il

Abstract. Permutation schedules in reentrant flow shops can be defined in several ways depending on job-passing and level-passing constraints across the sub-jobs. This gives rise to four permutation types. In this paper, we investigate the reentrant permutation flow shop with the objective of total completion time for the four permutation types. We show that allowing level passing is more effective at reducing the objective function value than allowing job passing. We conduct a systematic analysis of small instances to demonstrate this observation. We also address larger instances using the simulated annealing metaheuristic. The computational results demonstrate the strength of level passing with respect to the total completion time objective. In particular, the permutation type that allows level passing while prohibiting job passing offers a good trade-off for obtaining high-quality solutions in a reduced solution space.

Keywords: Permutation Flow Shop · Reentrant Flow Shop · Permutation Types · Total Completion Time · Simulated Annealing

1 Introduction

A flow shop is a scheduling environment in which all jobs visit a set of machines in the same order. A well-studied special case is the *permutation* flow shop (PFS), in which the job sequence is identical across all machines. In a standard flow shop, each job visits each machine at most once. However, in many real-world manufacturing systems, jobs must return to the same machine multiple times during processing. Such problems are referred to as *reentrant* flow shops (RFS). A particular class of RFS is the *cyclic* RFS, in which each job passes through all machines multiple times [3]. A cyclic RFS is characterized by n jobs, m machines, and l levels. Each level represents a full pass of a job through the machines in the order $(1, 2, \ldots, m)$. The processing of a job at a specific level is called a sub-job. Graves et al. [7] first introduced the RFS problem, and it has since been studied extensively, as reviewed by von Aspern et al. [1].

The notion of permutation has been extended to the RFS problem, leading to the *reentrant permutation flow shop* (RPFS), primarily in the cyclic setting.

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

In previous work [12, 13], we defined four RPFS permutation types based on two scheduling constraints:

- **Level passing:** whether a job may proceed to a higher level before all sub-jobs at the current level are completed. If so, level passing is allowed; otherwise, it is prohibited.
- **Job passing:** whether the job order within a given level must be identical across all levels. If so, job passing is prohibited; otherwise, it is allowed.

The four permutation types are defined according to whether level passing and job passing are allowed or prohibited: Passing Prohibited (PP), Level Passing Prohibited (LPP), Job Passing Prohibited (JPP), and Passing Allowed (PA). In particular, LPP prohibits level passing while allowing job passing, whereas JPP prohibits job passing while allowing level passing. In previous work, we showed that, for the makespan objective, the freedom afforded by allowing job passing is more effective than that provided by level passing [13]. In particular, LPP proved to be a successful permutation type in balancing the objective (high solution quality) and effort (low solution space).

Most previous studies on reentrant flow shop problems have focused on the makespan ($C_{\max}$) objective. Conversely, the *total completion time* objective has received considerably less attention, particularly in the RPFS setting. Here, we investigate the total completion time objective in cyclic RPFS under four permutation types. The problem is denoted by $F_m|\text{cyclic-reentrant}, \text{perm-}X|\sum C_j$, where $X \in \{\text{PP}, \text{LPP}, \text{JPP}, \text{PA}\}$, using the three-field notation [6]. We focus on small instances, for which optimal solutions can be obtained, analyze their properties, and examine how they differ from those under the makespan objective. In addition, we propose a simulated annealing (SA) metaheuristic to address larger instances.

Small illustrative instances show that neither level passing nor job passing uniformly dominates the other; depending on the processing times, each type of freedom may be either crucial or negligible. This was also the case for the makespan objective. Nevertheless, our computational study on both small and large instances reveals a clear overall trend: For the $\sum C_j$ objective, the flexibility associated with level passing tends to have a stronger impact than that associated with job passing. This contrasts with our findings for the makespan objective, where job passing was more influential. In particular, JPP (rather than LPP) provides a favorable trade-off between solution quality and computational efficiency. This can be intuitively explained by the nature of the $\sum C_j$ goal, which aggregates the completion times of all jobs and therefore favors schedules that enable earlier finishing of individual jobs. Such early finishing has little impact on the makespan, which is determined solely by the completion time of the final job.

The remainder of the paper is organized as follows. Subsection 1.1 reviews relevant literature on the total completion time objective in flow shop and RFS settings. Section 2 analyzes small instances, while Section 3 presents an SA–based approach for larger instances. A discussion in Section 4 concludes the paper.

1.1 Related Work

The PFS problem with the total completion time objective is NP-hard already for two machines [5]. Therefore, a variety of heuristics have been proposed, including the RZ heuristic [11], the NEH-based flowtime heuristic [4], and the LR heuristic [10]. The literature on cyclic RPFS under the total completion time objective remains limited. To the best of our knowledge, only a few closely related studies have considered adapting classical PFS heuristics to reentrant settings. One such study is of Xu et al. [15], who consider an RPFS with a position-based learning effect using a single job sequence across all levels, which can be interpreted as a structure corresponding to the PP type in our framework. They apply classical heuristics, including RZ and NEH, further enhanced by local search and genetic algorithm procedures. Jing et al. [9] studied a model equivalent to the PA structure and observed that, since the objective is to minimize the total completion time, it is beneficial to process a job's sub-jobs as close together as possible so that the job completes and leaves the system early. However, scheduling sub-jobs too closely may result in significant idle time due to precedence constraints between levels. Accordingly, they propose a heuristic algorithm based on a k-insertion technique, which iteratively improves sub-job sequences by controlling the distance between levels. For the two-machine case, they also derive a lower bound based on Johnson's rule. Aspern et al. [1] study RFS under the total weighted completion time objective, assuming unit processing times. They propose the Least Remaining Loops (LRL) rule, which is optimal for the unweighted case, and analyze its weighted extension. They also develop a pseudo-polynomial-time algorithm and a fully polynomial-time approximation scheme for a fixed number of machines. These works highlight different aspects of the total completion time objective in reentrant settings, yet a comprehensive treatment of cyclic RPFS across all permutation types remains largely unexplored.

2 Analysis of Small Instances

Following the approach for the makespan objective [13], we analyze the impact of the different permutation types on solution quality under the total completion time objective. Let PP, LPP, JPP, and PA, denote the solution space for the corresponding permutation types. Let $\sum C_j(OPT, X)$ denote the optimal total completion time for $X \in \{\text{PP}, \text{LPP}, \text{JPP}, \text{PA}\}$. By definition[1], $\text{PP} \subset \text{LPP}, \text{JPP} \subset \text{PA}$. Therefore, $\sum C_j(OPT, \text{PA}) \leq \sum C_j(OPT, \text{LPP}) \leq \sum C_j(OPT, \text{PP})$ and similarly $\sum C_j(OPT, \text{PA}) \leq \sum C_j(OPT, \text{JPP}) \leq \sum C_j(OPT, \text{PP})$. In some instances, equalities may replace some inequalities. When $\sum C_j(OPT, \text{PP}) = \sum C_j(OPT, \text{PA})$ for a given instance, we say that PP dominates PA for that instance, because PP achieved the same objective as PA even though PP is a strict subset of PA. Generalizing this, if $X \subset Y$ and $\sum C_j(OPT, X) = \sum C_j(OPT, Y)$, we say that X dominates Y. If, on the other hand, $\sum C_j(OPT, Y) < \sum C_j(OPT, X)$,

[1] The notation $\subset$ is used here to indicate a proper subset.

we say that Y dominates X. This dominance relation emphasizes the tension between quality and efficiency. An additional measure for the quality of a permutation type for a given instance is the optimality gap, defined as

$$\text{Optimality gap}(X) \;=\; \frac{\sum C_j(OPT, X) - \sum C_j(OPT, \text{PA})}{\sum C_j(OPT, \text{PA})} \times 100\,.$$

2.1 Simple Examples

Examples 1–4 below demonstrate that each of the four permutation types may dominate the other permutation types. In each example, every set among PP, LPP $\setminus$ PP (strict LPP), JPP $\setminus$ PP (strict JPP), and PA $\setminus$ (LPP $\cup$ JPP) (strict PA) admits a unique optimal solution.

Example 1. Consider a problem with 2 jobs, 2 machines, and 2 levels with the following processing times (each matrix line corresponds to a job, each column to a machine in a certain level, where different levels are separated by a large space):

$$\begin{pmatrix} 1 & M & 1\,1 \\ M & 1 & 1\,1 \end{pmatrix}$$

Assume that M is a sufficiently large number. The optimal schedule is a PP schedule, with objective value $\Sigma C_j = 2M + 7$. It is optimal already for $M = 2$. Figure 1 (left) shows the Gantt chart of the optimal schedule with $M = 10$. Different jobs are shown by different colors. Stars indicate the completion time of each job. Figure 1 (right) shows the sub-job order, emphasizing the nature of the permutation type.

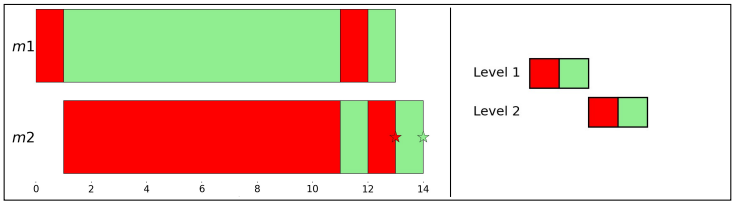


Fig. 1: Gantt chart (left) and sub-job order (right); PP-dominated.

In Example 1, PP dominates the other permutation types. Optimality can be proved by brute force, but it follows from the simultaneous processing of operations with long processing times. This is also true for the following examples of optimal schedules, which illustrate the respective dominance of LPP, JPP, and PA. These solutions are also optimal for the makespan objective [13].

Example 2. Consider a problem with 2 jobs, 2 machines, and 2 levels with the following processing times:

$$\begin{pmatrix} 1 & M & M & 1 \\ M & 1 & 1 & M \end{pmatrix}$$

The optimal schedule, shown in Figure 2, is a strict LPP schedule, with objective value $\sum C_j = 4M + 7$.

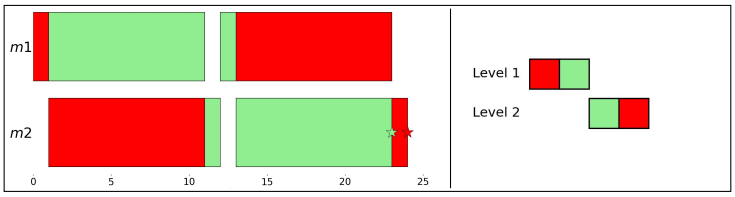


Fig. 2: Gantt chart (left) and sub-job order (right); LPP-dominated.

Example 3. Consider a problem with 3 jobs, 2 machines, and 2 levels with the following processing times:

$$\begin{pmatrix} 1 & M & 1 & M \\ M & 1 & 1 & M \\ M & 1 & M & 1 \end{pmatrix}$$

The optimal schedule, shown in Figure 3, is a strict JPP schedule, with objective value $\sum C_j = 8M + 9$.

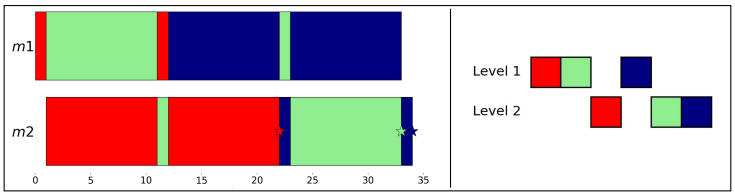


Fig. 3: Gantt chart (left) and sub-job order (right); JPP-dominated.

Example 4. Consider a problem with 3 jobs, 2 machines, and 2 levels with the following processing times:

$$\begin{pmatrix} 1 & 2M & 1 & M \\ 2M & 1 & 2M & 1 \\ M & 1 & 1 & 2M \end{pmatrix}$$

The optimal schedule, shown in Figure 4, is a strict PA schedule, with objective value $\sum C_j = 13M + 11$.

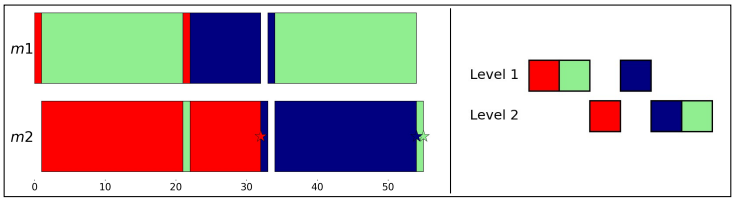


Fig. 4: Gantt chart (left) and sub-job order (right); PA-dominated.

2.2 Small Instances: Varying n and l

We consider 18 experimental sets, each containing 50 instances. Optimal solutions were obtained by exhaustive search for all instances, by enumerating all feasible permutations and evaluating their total completion times. The number of machines is fixed at $m = 5$, with n and l varying. The experimental sets include $n = 2$ with $l = 2, \ldots, 11$, $n = 3$ with $l = 2, \ldots, 5$, $n = 4$ with $l = 2, 3$, and $n = 5, 6$ with $l = 2$. We denote an experimental set by $n \times m \times l$. These instances were originally generated for experiments on the makespan objective [13]; details on their generation and full results can be found in that work. Nonetheless, we summarize key findings from the makespan experiments [13] for comparison with our total completion time results.

The Pareto-dominant experimental sets, those corresponding to the largest combinations of n and l, are $2 \times 5 \times 11$, $3 \times 5 \times 5$, $4 \times 5 \times 3$, and $6 \times 5 \times 2$. The results for these sets are presented in Figure 5 as Venn diagrams, one for each experimental set. For each set, we report the number of instances in which a solution space is dominant over the others. For example, among the $2 \times 5 \times 11$ instances, PP dominates in 15 cases, LPP in 9, JPP in 24, and PA in 2. The total may exceed 50 (in other experimental sets) because, in some instances, both LPP and JPP dominate. In addition to these counts shown in bold, we report in parentheses the size of the solution space (i.e., the number of feasible schedules) for each permutation type.

The results for all 18 experimental sets are presented in Figures 6 and 7, showing the effect of varying n and l, respectively. As n or l increases, the number of instances in which PP dominates decreases and shifts toward the other sets, initially to LPP, then increasingly to JPP, and eventually to PA. These findings differ from the makespan results, in which PP remained dominant more frequently and declined more gradually as n and l increased, with a stronger shift toward LPP and a weaker shift toward JPP.

Table 1 reports the average optimality gap across the Pareto-dominant experimental sets and the maximum optimality gap observed among the 50 instances of each set. The largest average optimality gap is observed in PP, followed by LPP and JPP. Compared to the makespan results, the average optimality gap for PP and LPP is higher, and for JPP it is lower. The largest average optimality gap observed in the set $6 \times 5 \times 2$ is 1.25% for the makespan objective, while it reaches 5.94% for the total completion time objective. The JPP type exhibits lower average optimality gaps than the LPP type across all examined sets, even

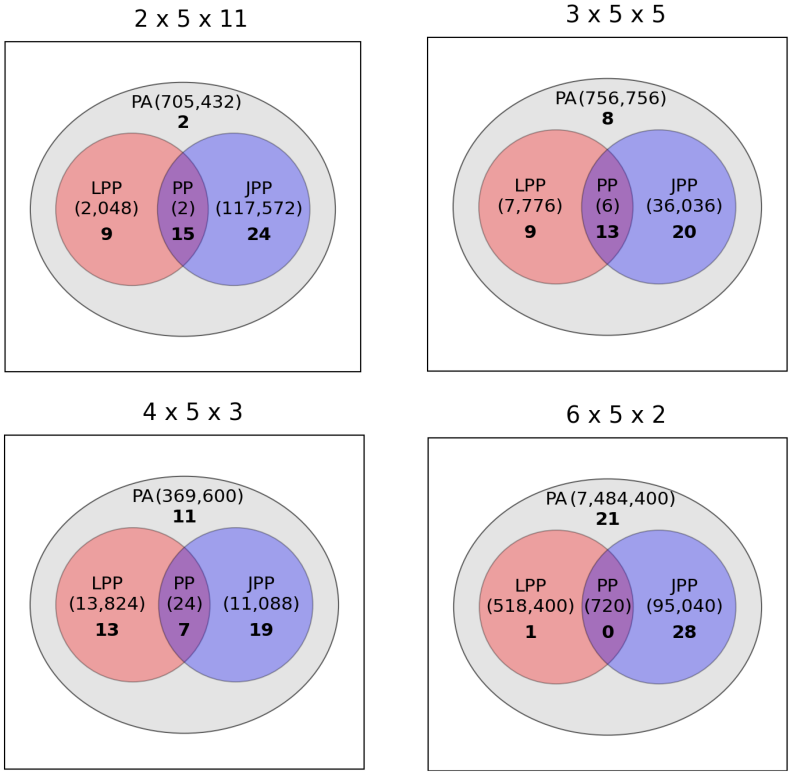
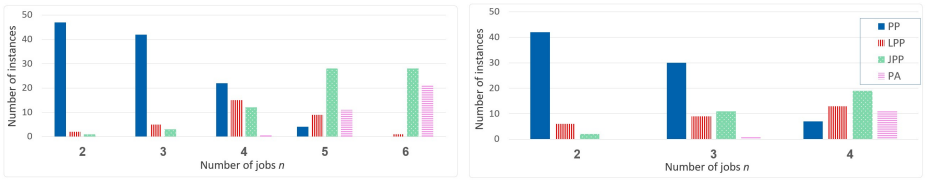


Fig. 5: Venn diagrams of the solution spaces showing the number of instances each space dominates (bold), and their sizes (parentheses).

in cases where its solution space is smaller. In general, the average optimality gap within each permutation type tends to increase as n or l increases.

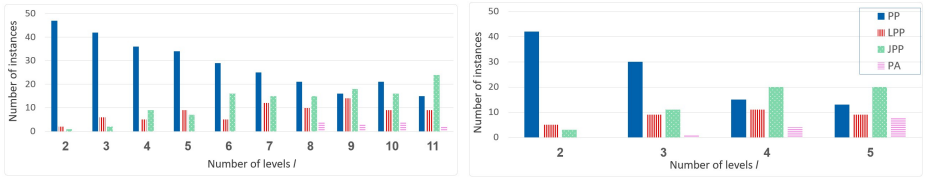
Another measure is the number of optimal solutions per instance, reported in Table 2 for the Pareto-dominant sets. For each permutation type, we report the average number of optimal solutions per instance and the percentage of instances with a single optimal solution. The average number of optimal solutions per instance is very small relative to the size of the solution space, and substantially lower than those observed for the makespan objective. For example, consider the PA type in the $6 \times 5 \times 2$ set. Its solution space contains approximately 7.5 million feasible schedules, while the average number of optimal solutions per instance is only 1.12, with 88% of the instances having a unique optimal solution.

These observations highlight the increasing difficulty of identifying optimal solutions as the problem size grows. For larger RPFS instances, exact solution methods become computationally impractical, and other approaches such as heuristics or metaheuristics are required. Based on our previous findings for the makespan objective, extensions of classical PFS heuristics to the RPFS setting



(a) $l = 2$ (b) $l = 3$

Fig. 6: Number of instances in which PP, LPP, JPP, or PA dominates, for $m = 5$, as a function of n.



(a) $n = 2$ (b) $n = 3$

Fig. 7: Number of instances in which PP, LPP, JPP, or PA dominates, for $m = 5$, as a function of l.

were shown to be less effective as the problem size increases. Therefore, rather than directly adapting such heuristics, we adopt a metaheuristic approach and apply simulated annealing (SA) to address the total completion time objective.

3 Metaheuristic Results

In this section, we show that trends observed in small instances also hold in larger ones. To systematically explore the solution space, we employ SA, which is an effective metaheuristic for exploring large search spaces and producing high-quality solutions to complex optimization problems. Specifically, SA has been widely used in flow shop scheduling, e.g., [2]. We previously considered an SA approach for RPFS to address the makespan objective [14]; here, we apply it to the $\sum C_j$ objective.

3.1 Simulated Annealing for RPFS

Three standard neighborhood operators from the PFS setting [8] were extended to the four RPFS types as described below. These include:

1. Adjacent swap: interchange two adjacent jobs.
2. Global swap: interchange two jobs at arbitrary positions.
3. Insert: remove a job and reinsert it elsewhere.

Their extensions to RPFS were:

	$2 \times 5 \times 11$		$3 \times 5 \times 5$		$4 \times 5 \times 3$		$6 \times 5 \times 2$	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max
PP	1.39	6.31	1.59	9.51	2.27	7.03	5.94	12.64
LPP	0.71	3.68	0.89	5.40	1.20	6.35	4.34	10.66
JPP	0.29	3.48	0.57	3.66	0.67	4.73	0.35	1.98

Table 1: Average and maximum optimality gaps (%).

	$2 \times 5 \times 11$		$3 \times 5 \times 5$		$4 \times 5 \times 3$		$6 \times 5 \times 2$	
	Avg	Unique (%)	Avg	Unique (%)	Avg	Unique (%)	Avg	Unique (%)
PP	1.00	100	1.00	100	1.04	96	1.10	90
LPP	1.06	94	1.38	72	1.54	70	2.30	50
JPP	1.06	94	1.16	90	1.16	86	1.12	88
PA	1.04	96	1.28	80	1.32	82	1.12	88

Table 2: Average number of optimal solutions per instance (Avg) and percentage of instances with a unique optimal solution (Unique).

- **PP:** Applied exactly as in PFS, since neither job passing nor level passing is allowed.
- **LPP:** Applied independently within each level, either sequentially across all levels (level-by-level) or on a randomly chosen level.
- **JPP:** Applied to the sub-job sequence while preserving precedence (level $k+1$ cannot precede level k) and maintaining a consistent job order across levels. To explore job orders, an additional adjacent swap was applied to the job sequence, inducing the corresponding reordering of all sub-jobs, yielding four neighborhood structures.
- **PA:** Applied to the sub-job sequence subject only to precedence constraints, providing the most flexible neighborhoods.

3.2 Algorithm Parameters and Move Selection

We used an initial temperature $T_0 = 100$, a final temperature $T_{\text{final}} = 0.1$, and a cooling rate $\alpha = 0.98$. At each temperature, $10 \cdot n$ neighbors were evaluated for the PP type and the level-by-level LPP, and $5 \cdot n \cdot l$ for the random-level LPP, JPP, and PA. The initial sequence was generated as a random permutation.

For PP, LPP, and PA, moves were chosen uniformly at random. For JPP, moves followed a (0.3, 0.3, 0.3, 0.1) distribution, with the job-order move assigned a lower probability due to its stronger structural effect. We applied a temperature-dependent scheme: at high temperatures, selection followed the base distribution; as temperature decreased, the probabilities of global swap and insert moves decreased linearly, while local moves increased, shifting from exploration to exploitation. The job-order move probability remained fixed (0.1).

3.3 Experimental Results

We first evaluate the performance of SA relative to the optimal solutions of the small instances (Subsection 2.2). Let $\sum C_j(\mathrm{SA}, X)$ denote the total completion time computed by SA for permutation type $X \in \{\mathrm{PP}, \mathrm{LPP}, \mathrm{JPP}, \mathrm{PA}\}$. The performance gap relative to optimal solutions is

$$\text{Performance gap}(X) = \frac{\sum C_j(\mathrm{SA}, X) - \sum C_j(\mathrm{OPT}, X)}{\sum C_j(\mathrm{OPT}, X)} \times 100.$$

The results are presented in Table 3. For each Pareto-dominant set, we report the percentage of instances for which SA attained the optimal objective function for each permutation type, along with the average and maximum performance gaps. The results indicate that the proposed SA approach performed effectively across all permutation types, consistently achieving high optimality rates and very small performance gaps. In particular, for the PP and LPP types, the method attained optimal solutions in nearly all instances. For the JPP and PA types, a slight performance degradation was observed for the $6 \times 5 \times 2$ set. Nevertheless, even in this case, the obtained solutions remained close to the optimal solutions, as reflected by the relatively small average and maximum performance gaps. These results suggest that the proposed SA mechanism effectively balances exploration and exploitation under the total completion time objective.

	$2 \times 5 \times 11$			$3 \times 5 \times 5$			$4 \times 5 \times 3$			$6 \times 5 \times 2$		
	Opt	Avg	Max	Opt	Avg	Max	Opt	Avg	Max	Opt	Avg	Max
PP	100	0.00	0.00	100	0.00	0.00	100	0.00	0.00	100	0.00	0.00
LPP	100	0.00	0.00	100	0.00	0.00	100	0.00	0.00	98	<0.01	0.10
JPP	100	0.00	0.00	100	0.00	0.00	98	0.01	0.53	90	0.15	2.25
PA	100	0.00	0.00	98	0.01	0.36	98	0.01	0.26	66	0.17	1.48

Table 3: SA performance: percentage of optimal solutions, average gap, and maximum gap (%).

Since the optimal solutions are unknown for large instances, performance is measured relative to the *best* solution found by SA for each instance. In this setting, methods with smaller search spaces may occasionally yield better solutions. Let $\sum C_j(\mathrm{SA}, \mathrm{BEST})$ denote the best (minimal) total completion time computed by SA for an instance. The performance gap is defined as

$$\text{Performance gap}(X) = \frac{\sum C_j(\mathrm{SA}, X) - \sum C_j(\mathrm{SA}, \mathrm{BEST})}{\sum C_j(\mathrm{SA}, \mathrm{BEST})} \times 100\,.$$

We generated 50 instances for a base configuration of $10 \times 5 \times 3$ and evaluated the performance gap. We then considered two types of variations: (i) varying the number of jobs $n \in \{5, 10, 15\}$ while keeping $m = 5$ and $l = 3$, and (ii) varying the number of levels $l \in \{2, 3, 4\}$ while keeping $n = 10$ and $m = 5$.

The obtained performance gaps are presented in Table 4. In the $5 \times 5 \times 3$ experiments, LPP attains the best solution in only 5 instances, including 2 ties with either PA or JPP. In all other instances across all experimental sets, the best solutions are obtained by JPP or PA. Moreover, the average gap of PP and LPP is consistently larger than that of JPP and PA, and increases with instance size, reaching values significantly higher than those observed for the makespan objective. Finally, JPP achieves performance comparable to PA, likely because its solution space strikes a balance: it is large enough to contain high-quality solutions, yet small enough to allow effective exploration by the SA search.

	$m = 5,\ l = 3$		
	$5 \times 5 \times 3$	$10 \times 5 \times 3$	$15 \times 5 \times 3$
PP	3.45	16.53	25.67
LPP	2.34	13.31	22.33
JPP	0.37	0.49	0.69
PA	0.42	1.15	0.57

(a) Varying n

	$n = 10,\ m = 5$		
	$10 \times 5 \times 2$	$10 \times 5 \times 3$	$10 \times 5 \times 4$
PP	15.05	16.53	16.80
LPP	11.97	13.31	13.70
JPP	1.03	0.49	0.45
PA	0.43	1.15	1.16

(b) Varying l

Table 4: Average performance gap (%) based on the base configuration $10 \times 5 \times 3$: (a) varying the number of jobs n, and (b) varying the number of levels l.

4 Discussion

The classification of permutation types highlights the role of two sources of flexibility: changes in job order across levels and passing between levels. While increased flexibility enlarges the search space and may improve solution quality, this is not guaranteed. In particular, both exact and heuristic approaches may benefit from restricting the search to smaller spaces, where high-quality solutions can sometimes be identified more effectively. The relative importance of these two types of flexibility differs between objectives. For makespan, improvements are mainly driven by allowing changes in job order across levels [13]. In contrast, for the total completion time objective considered in this work, the dominant effect arises from allowing jobs to overtake one another, enabling earlier completion without waiting for other jobs.

The computational results for small instances confirm that each solution space may be dominant. Nevertheless, dominance is observed more frequently for JPP, which often provides high-quality solutions even when it is not dominant over PA. This trend persists in the larger instances. In this setting, JPP often yields competitive and, in many cases, superior solutions.

These findings suggest that JPP provides an effective balance between flexibility and search complexity. Further computational study is needed to better understand the impact of problem parameters and to determine whether the observed trends reflect structural properties of the solution spaces or limitations of

the search. In particular, it is natural to ask whether the tendency to prioritize shorter jobs, which is well known in total completion time problems, also plays a significant role in the RPFS setting.

References

1. Aspern, M.v., Buld, F., Klein, N., Pinedo, M.: Flow shops with reentry: The total weighted completion time objective. Journal of Scheduling pp. 1–16 (2025)
2. Chen, R.M., Sandnes, F.E.: Flow shop scheduling based on a novel cooling temperature driven simulated annealing algorithm. Scientia Iranica. Transaction B, Mechanical Engineering **22**(4), 1545 (2015)
3. Emmons, H., Vairaktarakis, G.: Flow shop scheduling: theoretical results, algorithms, and applications, vol. 182. Springer Science & Business Media (2012)
4. Framinan, J.M., Leisten, R., Ruiz-Usano, R.: Comparison of heuristics for flowtime minimisation in permutation flowshops. Computers & Operations Research **32**(5), 1237–1254 (2005)
5. Garey, M.R., Johnson, D.S., Sethi, R.: The complexity of flowshop and jobshop scheduling. Mathematics of operations research **1**(2), 117–129 (1976)
6. Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.R.: Optimization and approximation in deterministic sequencing and scheduling: a survey. In: Annals of discrete mathematics, vol. 5, pp. 287–326. Elsevier (1979)
7. Graves, S.C., Meal, H.C., Stefek, D., Zeghmi, A.H.: Scheduling of re-entrant flow shops. Journal of operations management **3**(4), 197–207 (1983)
8. Hurkała, J., Hurkała, A.: Effective design of the simulated annealing algorithm for the flowshop problem with minimum makespan criterion. Journal of Telecommunications and Information Technology pp. 92–98 (2012)
9. Jing, C., Huang, W., Tang, G.: Minimizing total completion time for re-entrant flow shop scheduling problems. Theoretical Computer Science **412**(48), 6712–6719 (2011)
10. Liu, J., Reeves, C.R.: Constructive and composite heuristic solutions to the $p||\sum c_i$ scheduling problem. European Journal of Operational Research **132**(2), 439–452 (2001)
11. Rajendran, C., Ziegler, H.: An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs. European Journal of Operational Research **103**(1), 129–138 (1997)
12. Segal, I., Grinshpoun, T., Ilani, H., Shufan, E.: The meaning of permutation in reentrant permutation flow shop problems. In: PATAT 2024: 14th International Conference on the Practice and Theory of Automated Timetabling. pp. 99–109 (2024)
13. Segal, I., Grinshpoun, T., Ilani, H., Shufan, E.: Reentrant permutation flow shop problems: definitions and heuristics. Journal of Scheduling (2026). https://doi.org/10.1007/s10951-026-00873-4
14. Segal, I., Grinshpoun, T., Ilani, H., Shufan, E.: Using Simulated Annealing for Solving Reentrant Permutation Flow Shop Problems. In: Proceedings of the 20th International Workshop on Project Management and Scheduling (PMS 2026). pp. 66–69 (2026)
15. Xu, J., Lin, W.C., Wu, J., Cheng, S.R., Wang, Z.L., Wu, C.C.: Heuristic based genetic algorithms for the re-entrant total completion time flowshop scheduling with learning consideration. International Journal of Computational Intelligence Systems **9**(6), 1082–1100 (2016)

Improved bounds and solutions for the invigilator assignment problem

Reinout Annaert[1]

KU Leuven, Department of Computer Science, NUMA, Belgium
reinout.annaert@kuleuven.be

Abstract. This paper addresses the exam invigilator assignment problem with interaction-oriented objectives in a university with multiple departments. The problem consists of assigning invigilators to exams, where each exam requires a given number of invigilators, with each invigilator subject to availability and workload constraints. We consider two objective variants. Contact maximization seeks to maximize the number of unique interdepartmental contacts to stimulate interaction between departments. Contact reduction seeks to minimize the total number of unique contacts to limit health risks during pandemics or seasonal outbreaks. For both variants, we develop bounds on the objective value and propose a multi-neighborhood local search heuristic. Computational experiments on benchmark instances show that our proposed bounds improve or match almost all previously reported bounds. Moreover, the proposed heuristic improves upon all previously reported best-known solutions. For many contact-maximization instances, the best solution found matches the new bound, thereby proving optimality.

Keywords: Exam invigilation · Timetabling · Interaction-oriented objectives.

1 Introduction

The exam invigilator assignment problem (IAP) concerns the assignment of invigilators to a set of exams subject to operational requirements such as exam coverage, invigilator availability and workload restrictions. Despite its practical relevance in university examination planning, literature on the IAP remains relatively limited. Only recently, Verplancke et al. [15] introduced publicly available benchmark instances, thereby enabling far more systematic computational studies to take place.

Beyond operational feasibility, invigilator assignments can also be used to influence interactions between invigilators. Recent work has therefore considered interaction-oriented objectives, which aim to influence which invigilators supervise exams together. Assigning invigilators from different departments to the same exam stimulates interdepartmental interaction, whereas limiting the number of distinct contacts is desirable when contact reduction is important, for example during the COVID-19 pandemic or seasonal outbreaks such as flu.

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Verplancke et al. [15] formalize these two settings as contact maximization and contact reduction, and prove that both variants are NP-hard via reductions from the Social Golfer Problem [14] and Exact Cover by 3-Sets [5], respectively.

However, further opportunities remain to strengthen our computational understanding of these interaction-oriented variants. This paper addresses two important challenges: strengthening the best available bounds for both variants to enable a more accurate assessment of heuristic solution quality, and developing an improved local search heuristic.

The remainder of the paper is organized as follows. Section 2 reviews the related literature. Next, Section 3 formally defines the IAP with interaction-oriented objectives. Section 4 presents our new bounds for both variants, while Section 5 introduces our proposed local search heuristic. Section 6 reports the computational study. Finally, Section 7 concludes the paper.

2 Related work

The academic literature on the IAP is dominated by institution-specific case studies rather than benchmark-driven research. Across these studies, the focus is on constructing complete invigilation schedules for given examination timetables that respect specific institutional rules.

A common theme in the literature is the modeling of *fairness* and *preferences*. Fairness is often incorporated in the objective by minimizing dissatisfaction or balancing workload [1,4,8,9,10,12,13], or it can also be enforced through workload bounds or balancing constraints [3,11,15]. Preferences are treated similarly: some studies include them directly in the objective – through penalties for undesirable time slots, or preference-based assignment scores [1,8,9,12,13,15] – while other studies model them as constraints [3,4,11].

Beyond these modeling choices, previous research also differs in terms of the solution methods used. Several papers formulate the IAP as a mixed-integer programming (MIP) model [1,3,12,13,15]. These models are often combined with heuristic decomposition or variable-fixing techniques to obtain solutions within reasonable computation times. Other studies rely on constructive heuristics [9,10,11], goal programming [4] or heuristics such as scatter search [8] to generate solutions.

Despite this methodological variety, little attention has been paid to computing bounds, making it difficult to accurately assess solution quality. Moreover, the lack of public benchmark data has limited comparability, as most studies rely on proprietary instances from a single institution. As a result, meaningful comparisons between methods have remained scarce. Verplancke et al. [15] address this limitation by introducing public benchmark instances for the IAP, thereby enabling systematic comparison.

3 Problem description

The IAP with interaction-oriented objectives involves a set of exams E, invigilators I, timeslots T and departments K. Each exam $e \in E$ is scheduled during a timeslot $t_e \in T$ and requires d_e invigilators, where $d_e \in \mathbb{Z}_{\geq 0}$.

Each invigilator $i \in I$ is available during a subset of timeslots $A_i \subseteq T$ and belongs to a department $k_i \in K$. Furthermore, each invigilator must be assigned to a number of exams between a minimum and a maximum workload:

$$\underline{w}_i,\ \overline{w}_i \in \mathbb{Z}_{\geq 0}, \qquad \underline{w}_i \leq \overline{w}_i, \tag{1}$$

where $\underline{w}_i$ and $\overline{w}_i$ denote the minimum and maximum number of exams to which invigilator i must be assigned, respectively.

A feasible solution assigns exactly d_e available invigilators to each exam $e \in E$. An invigilator cannot be assigned to more than one exam in the same timeslot, and all workload bounds must be satisfied.

Interaction is modeled at two levels: *co-assignments* and *contacts*. Two distinct invigilators i and j are *co-assigned* to exam e if both are assigned to e. They have a *contact* if they are co-assigned to at least one exam. Thus, a contact is a unique unordered pair $\{i, j\}$, counted at most once regardless of how many exams invigilators i and j are co-assigned to. A contact $\{i, j\}$ is considered a *interdepartmental contact* if $k_i \neq k_j$.

We study two variants of the IAP. In contact maximization (IAP-CM), the goal is to maximize the number of interdepartmental contacts. In contact reduction (IAP-CR), the goal is to minimize the total number of contacts, regardless of whether they are interdepartmental or intradepartmental.

4 Improved bounds

To derive new bounds, we will use the following binary variables for assignments and contacts. Let $x_{i,e}$ be 1 if invigilator $i \in I$ is assigned to exam $e \in E$, and 0 otherwise. For each pair of invigilators $i, j \in I$ with $i < j$ and exam $e \in E$, let $w_{ij,e}$ be 1 if i and j are co-assigned to exam e, and 0 otherwise, and let y_{ij} be 1 if they have a contact, and 0 otherwise.

Using this notation, the objective of IAP-CM is

$$\max \sum_{\substack{i,j \in I \\ i<j,\ k_i \neq k_j}} y_{ij}, \tag{2}$$

whereas the objective of IAP-CR is

$$\min \sum_{\substack{i,j \in I \\ i<j}} y_{ij}. \tag{3}$$

4.1 Upper bounds for IAP-CM

The demand-only upper bound, introduced by [15], is based solely on the number of invigilators needed for an exam. For an exam e with demand d_e, at most $\binom{d_e}{2}$ unordered invigilator pairs can be co-assigned. Since each interdepartmental contact requires at least one co-assignment to an exam, the total number of interdepartmental contacts cannot exceed the total number of co-assigned pairs across all exams:

$$\sum_{e \in E} \binom{d_e}{2}. \tag{4}$$

To obtain a stronger bound, we enforce an upper bound on objective (2) by counting all interdepartmental co-assignments instead of all interdepartmental contacts. Since each contact corresponds to at least one co-assignment, this yields a valid upper bound. For each exam $e \in E$ and department $k \in K$, let us define

$$z_{k,e} = \sum_{i:\, k_i = k} x_{i,e}. \tag{5}$$

Thus, $z_{k,e}$ denotes the number of invigilators from department k assigned to exam e. For exam e, the total number of co-assigned pairs equals $\binom{d_e}{2}$, while the number of intradepartmental co-assigned pairs equals $\sum_{k \in K} \binom{z_{k,e}}{2}$. This yields the objective used to compute the upper bound:

$$\max \sum_{e \in E} \left(\binom{d_e}{2} - \sum_{k \in K} \binom{z_{k,e}}{2} \right). \tag{6}$$

Maximizing (6) is equivalent to minimizing the number of intradepartmental co-assigned pairs. Let us define:

$$C^\star = \min \sum_{e \in E} \sum_{k \in K} \binom{z_{k,e}}{2}. \tag{7}$$

Then the optimal value of the objective equals:

$$\mathrm{UB}_2 = \sum_{e \in E} \binom{d_e}{2} - C^\star. \tag{8}$$

Minimization problem (7) can be solved using a min-cost flow network. The network sends flow from invigilators to exams through timeslots and departments. A feasible flow corresponds to a solution satisfying workload, availability and exam demand constraints. Costs are incurred only on department–exam arcs and represent intradepartmental co-assigned pairs.

To model the network, we introduce the following auxiliary quantities. For each timeslot $t \in T$, let

$$E_t = \{e \in E : t_e = t\}, \tag{9}$$

and set $\delta_t = 1$ if $E_t \neq \emptyset$, and 0 otherwise.

We define the effective maximum workload of invigilator i as follows:

$$\overline{w}_i^{\text{eff}} = \min\left\{\overline{w}_i,\ \sum_{t\in A_i}\delta_t\right\} \qquad \forall i \in I. \tag{10}$$

For each department $k \in K$ and timeslot $t \in T$, let

$$n_{k,t} = \left|\{\, i \in I : k_i = k,\ t \in A_i \,\}\right| \tag{11}$$

denote the number of invigilators from department k available during timeslot t. For each exam $e \in E$ and department $k \in K$, let us define

$$m_{k,e} = \min\{d_e,\ n_{k,t_e}\} \tag{12}$$

as the maximum number of invigilators from department k that can be assigned to exam e.

The network's node types are summarized in Table 1. Let $D = \sum_{e\in E} d_e$ denote total exam demand. Node s represents the source supplying the D units of flow corresponding to invigilator assignments, while node f represents the sink collecting those D units of flow required to satisfy exam demand. For each invigilator $i \in I$, node v_i aggregates the assignments of invigilator i. For each available invigilator–timeslot pair (i, t), node $u_{i,t}$ ensures that invigilator i can be assigned to at most one exam during timeslot t. Nodes $p_{k,e}$ aggregate assignments of invigilators from department k to exam e, while node q_e collects all invigilators assigned to exam e. The arc types are summarized in Table 2, where ℓ and u denote the lower and upper bounds for each arc.

Table 1. Node types in the min-cost flow network.

Node	Exists for
s, f	–
v_i	$\forall i \in I$
$u_{i,t}$	$\forall i \in I,\ \forall t \in A_i,\ E_t \neq \emptyset$
$p_{k,e}$	$\forall e \in E,\ \forall k \in K :\ n_{k,t_e} > 0$
q_e	$\forall e \in E$

If $z_{k,e}$ invigilators from department k are assigned to exam e, they form $\binom{z_{k,e}}{2}$ intradepartmental pairs. This is modeled using $m_{k,e}$ parallel arcs from $p_{k,e}$ to q_e with incremental costs $0, 1, \ldots, m_{k,e} - 1$, so that sending $z_{k,e}$ units of flow incurs the following cost:

$$\sum_{j=1}^{z_{k,e}}(j-1) \ =\ \binom{z_{k,e}}{2}. \tag{13}$$

Solution method. We solve the min-cost flow problem with lower and upper bounds by applying the standard transformation that eliminates lower bounds

Table 2. Arc types in the min-cost flow network.

Arc	Exists for	ℓ	u	Cost
$(s \to v_i)$	$\forall i \in I$	$\underline{w}_i$	$\overline{w}_i^{\text{eff}}$	0
$(v_i \to u_{i,t})$	$\forall i \in I,\ \forall t \in A_i,\ E_t \neq \emptyset$	0	1	0
$(u_{i,t} \to p_{k_i,e})$	$\forall (i,t),\ \forall e \in E_t$	0	1	0
$(p_{k,e} \to q_e)$	$\forall e \in E,\ \forall k \in K,\ j = 1, \ldots, m_{k,e}$	0	1	$j-1$
$(q_e \to f)$	$\forall e \in E$	d_e	d_e	0

and introduces node balances. Let (p, q) denote an arc in the network and let b_v denote the balance of node v, that is, the net outflow at node v. Initially, the balances are $b_s = D$, $b_f = -D$, and $b_v = 0$ for all other nodes. The transformation then proceeds as follows:

1. Replace each arc capacity with $u_{pq} - \ell_{pq}$.
2. Adjust node balances $b_p \leftarrow b_p - \ell_{pq}$ and $b_q \leftarrow b_q + \ell_{pq}$.
3. Add arc (f, s) with sufficiently large capacity and zero cost.
4. Introduce an auxiliary super-source and super-sink. Connect nodes with $b_v < 0$ to the super-source and nodes with $b_v > 0$ to the super-sink.

We then solve the resulting minimum-cost flow problem using successive shortest augmenting paths with reduced costs. All capacities and balances are integral, hence an optimal integral flow exists. The optimal flow value equals Equation (7), thereby yielding upper bound UB_2.

4.2 Lower bounds for IAP-CR

Let us begin by recalling the lower bound proposed by [15] for IAP-CR. Their bound is based on two observations. First, supervising an exam with demand d_e creates exactly $\binom{d_e}{2}$ co-assigned invigilator pairs, while assigning invigilators in a new timeslot may force additional contacts with invigilators encountered earlier. Second, any feasible solution must use a sufficiently large number of distinct invigilators to cover total demand while satisfying workload constraints. The bound is constructed by processing timeslots iteratively. Given the timeslots already processed, each step selects the next timeslot so as to maximize the increase in the bound, given the timeslots already processed. This yields a valid lower bound, which we denote by LB_1:

$$\sum_{\substack{i,j \in I \\ i<j}} y_{ij} \ \geq \ \text{LB}_1. \tag{14}$$

To derive an alternative lower bound, we compare the total number of co-assigned pairs that must be created when supervising the exams with the maximum number of co-assignments that can be realized through a single contact. This yields a covering bound over contacts.

Supervising exam e with demand d_e creates exactly $\binom{d_e}{2}$ unordered co-assigned invigilator pairs. Hence any feasible solution induces a total of Equation (4) co-assigned pairs across all exams.

We can bound the contribution of a single contact. Indeed, an invigilator can be assigned to at most one exam per timeslot, while a pair $\{i, j\}$ can be co-assigned to at most one exam per timeslot during which both invigilators are available. Workload bounds also limit how often each invigilator can be assigned. We define the capacity of contact $\{i, j\}$ as follows:

$$u_{ij} = \min \left\{ \sum_{t \in A_i \cap A_j} \delta_t,\ \overline{w}_i^{\text{eff}},\ \overline{w}_j^{\text{eff}} \right\} \qquad \forall\, i, j \in I :\ i < j. \tag{15}$$

This quantity is an upper bound on the number of co-assignments that contact $\{i, j\}$ can realize in any feasible solution. If $\mathcal{M} = \{\{i, j\} : y_{ij} = 1\}$ denotes the set of contacts in a feasible solution, then the total number of co-assignments generated by these contacts is at most

$$\sum_{\{i,j\} \in \mathcal{M}} u_{ij}. \tag{16}$$

Feasibility therefore requires that these capacities cover all co-assigned pairs induced by the exams, namely:

$$\sum_{\{i,j\} \in \mathcal{M}} u_{ij} \ \geq\ \sum_{e \in E} \binom{d_e}{2}. \tag{17}$$

Let CoverLB(r) denote the minimum number of contacts whose capacities sum to at least r. Since each contact contributes at most u_{ij} co-assignments, every feasible solution of the IAP-CR must satisfy

$$\sum_{\substack{i,j \in I \\ i<j}} y_{ij} \ \geq\ \text{CoverLB}\left(\sum_{e \in E} \binom{d_e}{2} \right). \tag{18}$$

We therefore define the lower bound as follows:

$$\text{LB}_2 = \text{CoverLB}\left(\sum_{e \in E} \binom{d_e}{2} \right). \tag{19}$$

Solution method. Computing LB_2 reduces to selecting the smallest number of contacts whose capacities u_{ij} cover $\sum_{e \in E} \binom{d_e}{2}$. Since all contacts have unit cost, this problem is solved optimally by sorting values u_{ij} in nonincreasing order and selecting them until the required capacity is reached.

5 Local search heuristic

To quickly compute solutions for both the IAP-CM and IAP-CR, we will develop a multi-neighborhood local search heuristic. Its purpose is to converge quickly to high-quality solutions and ultimately improve upon the best-known results for the benchmark instances.

5.1 Initial solution

We start the search from an empty solution in which no invigilators are assigned to exam slots. In this initial state, exam demands are unmet and invigilator workloads may fall below their required lower bounds. To allow the search to start from this infeasible state, violations of constraints are penalized in the objective function with large penalty terms. As a result, the local search gradually repairs the infeasible solution during the search.

5.2 Neighborhoods

The local search procedure uses a diverse set of neighborhoods. At each iteration, one neighborhood is selected at random and a candidate move is generated. Candidate moves are evaluated using the Late-Acceptance Hill Climbing (LAHC) acceptance criterion [2], with history length H. Some neighborhoods perform small local changes, while others reconstruct larger parts of the solution.

We can group each of the neighborhoods into one of three categories: reassignment, swap and reconstruction. Reassignment neighborhoods modify a single assignment, swap neighborhoods exchange assignments between exams, and reconstruction neighborhoods rebuild larger parts of the solution.

Reassignment neighborhoods

Simple reassign. The invigilator assigned to an exam is replaced by another invigilator who is available for the corresponding timeslot.

Contact-guided reassign. The invigilator assigned to an exam is replaced by another invigilator based on the objective. The available candidate that maximizes interdepartmental contacts (or minimizes contacts) is selected.

Swap neighborhoods

Timeslot swap. Two invigilators assigned to different exams in the same timeslot are swapped.

Cross-timeslot swap. Two invigilators assigned to different exams in different timeslots are swapped.

Three-way swap. Three invigilators assigned to three different exams in the same timeslot are cyclically swapped.

Contact-guided swap. All possible swaps between pairs of invigilators assigned to different exams in the same timeslot are considered as candidate moves. Each

candidate swap is evaluated based on the contacts consequently created with the other invigilators.

Reconstruction neighborhoods

Ruin-and-recreate. All invigilators assigned to an exam are removed, after which the exam is greedily reconstructed.

Within-timeslot JV. One invigilator is removed from each exam in a randomly selected set of exams taking place during the same timeslot. The removed invigilators are then reassigned to the vacant positions by solving a linear assignment problem using the Jonker–Volgenant (JV) algorithm [6]. The number of reassigned invigilators, and thus the size of the assignment matrix, is limited by parameter $S_{\max}$.

Cross-timeslot JV. One invigilator is removed from each exam in a randomly selected set of exams. The removed invigilators are then reassigned to the vacant positions by solving a linear assignment problem using the JV algorithm, again limited to at most $S_{\max}$ reassigned invigilators.

6 Computational experiments

We evaluate our approach on the benchmark instances introduced by [15]. The experiments were performed on a machine with an AMD Ryzen 7 7800X3D processor and 32GB of RAM. We implemented the local search heuristic in Rust and compiled it in release mode.

We tuned two algorithm parameters automatically using the `irace` package [7]: history length H of the Late-Acceptance Hill Climbing acceptance criterion and the maximum size $S_{\max}$ of the JV assignment matrix. The tuning procedure consisted of 300 runs on all 17 instances and selected $H = 20$ and $S_{\max} = 15$, which we used in all experiments.

We compare our bounds and solutions against those of Verplancke et al. [15], who propose a multi-stage heuristic that combines a MIP model with two improvement phases. They first construct an initial solution by solving a polynomial-time MIP model for a preference-based variant of the IAP, using modified objective coefficients to indirectly incentivize either contact maximization or contact reduction. They then improve this solution through a swapping phase. Finally, they apply a fix-and-optimize heuristic in which they iteratively relax subsets of variables (such as those related to exams, timeslots, or invigilators) while fixing the remaining variables, before reoptimizing the resulting subproblems.

Since both approaches are stochastic, we performed five independent runs per instance for each approach, with a time limit of 600 seconds. For the approach of [15], we used the publicly available source code provided by the authors. We report both the average and the best objective value obtained across these runs.

The results for the IAP-CM in Table 3 show that our new upper bound UB_2 is significantly stronger than the previously reported bound UB_1. More precisely,

UB_2 improves UB_1 for 15 of the 17 instances and matches it for the remaining two (`ITC3` and `UG4`). Our heuristic matches or improves all previously reported best-known solutions. Indeed, for 15 of the 17 instances, the best solution found matches UB_2, thereby proving optimality. The results are also highly stable: all five runs attain the best reported value for 16 of the 17 instances.

Table 3. Comparison of bounds and solution methods for the IAP-CM variant.

Inst.	UB_1	UB_2	Obj_1		Obj_2		
			Avg	Best	Avg	Best	
ITC1	2,992	2,457	2,434.0	2,438	2,457.0	2,457	⊛
ITC2	4,172	4,137	4,106.2	4,107	4,137.0	4,137	⊛
ITC3	4,558	4,558	4,524.8	4,528	4,558.0	4,558	⊛
ITC4	1,377	1,325	1,310.6	1,314	1,325.0	1,325	⊛
ITC5	5,110	4,871	4,848.0	4,848	4,871.0	4,871	⊛
ITC6	2,030	1,912	1,892.6	1,895	1,912.0	1,912	⊛
ITC7	9,257	8,175	8,067.2	8,070	8,172.4	8,175	⊛
ITC8	5,015	4,262	4,234.6	4,240	4,262.0	4,262	⊛
ITC9	1,386	1,133	1,129.8	1,131	1,133.0	1,133	⊛
ITC10	1,791	1,524	1,523.4	1,524	1,524.0	1,524	⊛
ITC11	12,026	11,079	11,012.8	11,013	11,079.0	11,079	⊛
ITC12	988	779	779.0	779	779.0	779	⊛
UG1	79	78	78.0	78	78.0	78	⊛
UG2	1,603	1,507	1,477.0	1,477	1,507.0	1,507	⊛
UG3	1,178	1,098	1,079.0	1,087	1,097.0	1,097	
UG4	704	704	703.4	704	704.0	704	⊛
UG5	1,593	1,495	1,468.4	1,474	1,493.0	1,493	

UB_1, Obj_1: [15]; UB_2, Obj_2: this paper; ⊛ proven optimum.

Table 4. Comparison of bounds and solution methods for the IAP-CR variant.

Inst.	LB_1	LB_2	Obj_1		Obj_2	
			Avg	Best	Avg	Best
ITC1	224	365	1,669.0	1,651	1,285.2	1,274
ITC2	478	786	2,647.2	2,637	2,206.8	2,187
ITC3	773	1,140	3,228.0	3,225	2,749.2	2,745
ITC4	336	459	989.6	979	726.4	718
ITC5	648	900	2,898.4	2,896	2,659.6	2,629
ITC6	697	677	1,340.4	1,336	1,111.0	1,100
ITC7	727	1,277	5,440.2	5,429	5,132.8	5,076
ITC8	445	773	3,091.0	3,080	2,552.0	2,507
ITC9	329	347	910.6	903	747.6	726
ITC10	328	359	964.6	950	855.6	835
ITC11	1,492	1,718	6,731.4	6,697	6,443.6	6,402
ITC12	658	494	789.8	782	752.0	732
UG1	17	27	39.8	38	39.6	38
UG2	219	401	875.4	866	717.4	679
UG3	147	295	557.6	522	553.6	517
UG4	134	176	380.4	371	314.2	311
UG5	228	399	862	822	727.6	708

UB_1, Obj_1: [15]; UB_2, Obj_2: this paper; ⊛ proven optimum.

For the IAP-CR, Table 4 shows that our new lower bound LB_2 improves upon LB_1 for 15 of the 17 instances, often by a significant margin. The exceptions are `ITC6` and `ITC12`, where LB_1 remains stronger. Our heuristic improves the best-known solution for 16 of the 17 instances. Compared with IAP-CM, the results for the IAP-CR exhibit larger gaps between bounds and best solutions, as well

as larger differences between average and best run values. This indicates that the minimization variant is more computationally challenging for both bounding and local search. We hypothesize that this is due to the fact that department information cannot be exploited in this variant.

Figures 1 and 2 show the aggregated convergence behavior of the proposed heuristic compared to [15]. For each instance, both methods were run five times with time limits of 10 seconds, 1 minute, 2 minutes, 10 minutes, 20 minutes and 1 hour. The reported gap is computed with respect to the best-known solution for each instance. The solid line represents the average gap over time, while the shaded band indicates one standard deviation. For the IAP-CM, our method

Fig. 1. Average gap over time (IAP-CM).

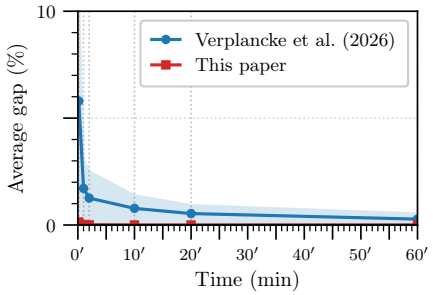
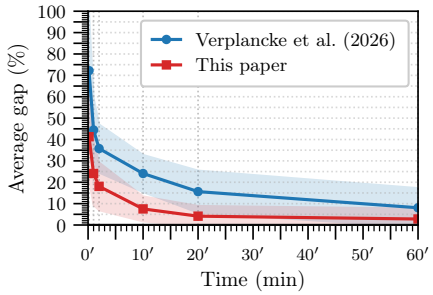


Fig. 2. Average gap over time (IAP-CR).



exhibits a very rapid improvement of the objective within the first few seconds. In fact, all instances except `ITC3` and `ITC11` reach a gap of 0% within 10 seconds. After one minute, all instances have a gap of 0% across all five runs. In contrast, the approach of [15] reduces the gap much more gradually and requires substantially more time to reach solutions of comparable quality.

For the IAP-CR, a different trend can be observed. Convergence is more gradual and takes longer, once again confirming that the minimization variant is more challenging for the heuristic. Nevertheless, our approach consistently outperforms [15], achieving both faster improvements in the early stages and better final solution qualities.

7 Conclusions and future directions

This paper strengthens the computational understanding of the invigilator assignment problem with interaction-oriented objectives. For both interdepartmental contact maximization and contact reduction variants, we (i) improve most existing bounds and (ii) improve the previously reported best-known solutions on the benchmark instances.

Our computational results also highlight a clear difference between the two variants. For the IAP-CM, our new upper bound is strong and the heuristic performs very well, enabling optimality to be proved for 15 out of 17 instances. For the IAP-CR, the proposed lower bound and heuristic solutions lead to significant improvements in both bounds and solution quality, but the gaps that remain indicate that this variant is more computationally challenging.

Several directions for future research remain. First, it would be worth investigating which instance characteristics drive computational difficulty in both variants. In particular, for the IAP-CM it would be interesting to study how the distribution of invigilators over departments affects both the quality of the bounds and the difficulty of finding good solutions. A second direction for future research is to investigate more flexible problem variants in which workload bounds and exam demands are treated as soft constraints rather than strict feasibility requirements.

Acknowledgments The author would like to thank Greet Vanden Berghe and Pieter Smet for proofreading, and Connolly Editorial for editorial consultation.

References

1. Al-Yakoob, S.M., Sherali, H.D., Al-Jazzaf, M.: A mixed-integer mathematical modeling approach to exam timetabling. Computational Management Science 7(1), 19–46 (2010)
2. Burke, E.K., Bykov, Y.: The late acceptance hill-climbing heuristic. European Journal of Operational Research 258(1), 70–78 (2017)
3. Güler, M.G., Geçici, E., Köroğlu, T., Becit, E.: A web decision support system for examination timetabling. Expert Systems with Applications 183, 115363 (2021)
4. Hanum, F., Romliyah, M.A., Bakhtiar, T.: Exam invigilators assignment problem: a goal programming approach. Applied Mathematical Sciences 9(58), 2871–2880 (2015)
5. Garey, M.R., Johnson, D.S.: Computers and intractability: A guide to the theory of NP-completeness. W.H. Freeman (1979)
6. Jonker, R., Volgenant, A.: Improving the Hungarian assignment algorithm. Operations Research Letters 5(4), 171–175 (1986)
7. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives 3, 43–58 (2016)
8. Lourenço, H.R., Martí, R., Laguna, M.: Assigning proctors to exams with scatter search. Universitat Pompeu Fabra (2001)
9. Matcı, D.K., Acar, I.: Modelling and optimization of the exam invigilator assignment problem based on preferences. Turkish Journal of Engineering 3(3), 140–148 (2019)
10. Mohmad Kahar, M.N., Kendall, G.: Universiti Malaysia Pahang examination timetabling problem: scheduling invigilators. Journal of the Operational Research Society 65(2), 214–226 (2014)
11. Othman, J., Wahab, A., Kadar, R., NizamWarris, S.: Assigning Examination Invigilator's Schedule Problem: An Approach of Programming Techniques Using Simple Sequential Assignment. JCRINN 2, 1–13 (2017)
12. Ozturk, Z.K., Ozturk, G., Sagir, M.: An automated multi-objective invigilator-exam assignment system. International Journal of Information Technology & Decision Making 9(2), 223–238 (2010)
13. Sagir, M., Ozturk, Z.K.: Exam scheduling: Mathematical modeling and parameter estimation with the Analytic Network Process approach. Mathematical and Computer Modelling 52(5–6), 930–941 (2010)
14. Triska, M.: Solution methods for the social golfer problem. Master's thesis, TU Wien, Austria (2008)
15. Verplancke, H., Van Bulck, D., Limère, V., Schmid, N.A.: The (a) social exam invigilator assignment problem. Journal of Scheduling (2026)

Part II

Extended Abstracts and Demos

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Evaluating Predict-then-Optimize methods for surgical case planning

Pieter Smet[1] and David Ajit Kirpekar-Sauer[2]

[1] KU Leuven, Department of Computer Science, Gebroeders de Smetstraat 1, 9000 Gent, Belgium
[2] Technical University of Denmark, Anker Engelundsvej, 2800 Kongens Lyngby, Denmark

1 Introduction

We consider a Surgical Case Planning (SCP) problem in which patients are selected from a waiting list for elective surgery and assigned surgery dates. We account for the limited capacity of two key resources: operating rooms (ORs) and anesthesiologists who must attend each procedure [1]. This problem is solved after both the master surgery schedule (MSS) of the ORs and the rosters for the anesthesiologist have been determined, but before the patients' actual clinical conditions and potential staff absences are known. As a result, the SCP decisions are subject to significant uncertainty [7]. We model this uncertainty through two problem parameters that are unknown at the time of solving: (i) surgery durations and (ii) the absence or presence of scheduled anesthesiologists.

To address this uncertainty, we adopt a Predict-then-Optimize (PTO) approach. In the first stage, two prediction models are used to predict surgery durations and anesthesiologist absences. These predictions are then used in the second stage, where a deterministic optimization model is solved to generate a solution for the SCP problem. It is well known that for some problems, improving prediction accuracy does not necessarily lead to better solutions of the optimization model [4]. The relationship between prediction accuracy and solution quality becomes even more complicated when two prediction models are involved, as is the case in the SCP problem. In this work, we empirically evaluate this relationship, thereby enabling an analysis of how solution quality changes as a function of prediction accuracy.

2 Problem description

For a given planning period, the considered SCP problem involves selecting a subset of patients from a waiting list and assigning each of them a feasible surgery date, OR and anesthesiologist, while respecting the capacity constraints of ORs and anesthesiologists. The anesthesiologist rosters and MSS specify the available hours for both of these resources on each day of the planning period. All patients on the waiting list are eligible for surgery in the planning period. Each patient has a due date before which they must be operated on, as well as

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

an estimated surgery duration. Additionally, clinical requirements may restrict compatibility between patients and certain anesthesiologists or ORs. For example, certain patients require anesthesiologists with specific skills, while certain procedures depend on equipment that is not available in all ORs.

Two problem parameters are unknown at the time of solving:

1. The surgery duration t_p of patient p. Although historical averages are often used in practice, these may be inaccurate. We assume instead that this duration can be predicted using patient-specific features $\mathbf{z}_p$, such as their age, clinical condition and medical history [5].
2. The presence of anesthesiologist a scheduled to work on a day d, represented as a binary parameter b_{ad}. Unexpected absenteeism may occur due to illness or external factors. We assume that absences can be predicted from anesthesiologist-specific features $\mathbf{z}_a$, such as workload intensity, recent absenteeism frequency and seasonal illness trends [3].

The objective is to minimize a weighted sum of two objectives: (i) the number of external workers required to cover for absent anesthesiologists and (ii) the total waiting time of all patients. Unscheduled patients are carried over to the next planning period, incurring additional waiting time.

3 Methodology

To solve the SCP problem, we apply the PTO approach illustrated in Figure 1. Two prediction tasks are carried out: a regression task to predict surgery duration $\hat{t}_p$, and a binary classification task to predict anesthesiologist absences $\hat{b}_{ad}$. The optimization model is then solved using integer programming to obtain an optimal solution $x^*(\hat{t}_p, \hat{b}_{ad})$ from the set of feasible solutions $\mathcal{X}$.

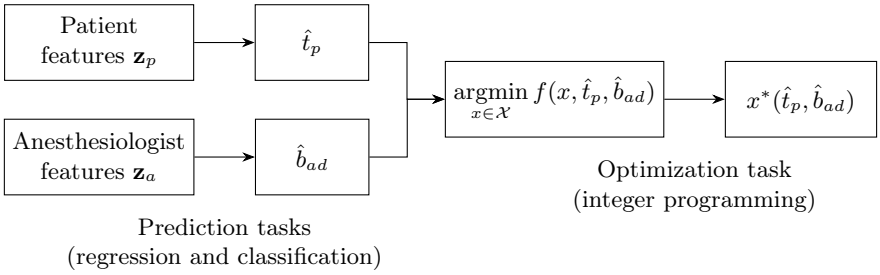


Fig. 1. Schematic overview of the PTO approach.

We conduct a series of experiments to analyze how the accuracy of the regression and classification models impacts solution cost. To this end, we use the algorithms introduced in [2,6] to simulate the predictions that models would produce at a specified level of accuracy. By systematically adjusting this accuracy,

we obtain an empirical mapping between prediction performance metrics and solution cost. We vary the prediction accuracies of both models simultaneously to assess interaction effects between the two sources of uncertainty.

Solution cost is computed by simulating the realization of the actual surgery durations and anesthesiologist absences given a solution $x^*(\hat{t}_p, \hat{b}_{ad})$ obtained with predictions. Once the uncertain parameters are revealed on each day, we apply one of several rescheduling policies to restore feasibility, if necessary. These policies may involve postponing surgeries, reassigning them to a different OR or anesthesiologist, or a combination of these actions. Each change incurs a cost, which is minimized during the rescheduling process.

Preliminary computational results based on data from a Danish hospital indicate that KPIs such as the number of postponed, canceled or resource-changed patients, are affected in a similar way by the accuracy of surgery duration predictions. Notably, the results demonstrate that improvements in prediction accuracy do not necessarily lead to better KPI values. Furthermore, the accuracy of anesthesiologist absence predictions appears to have only a limited influence on these results. Overall, these findings suggest that the relationship between prediction accuracy and operational performance is not straightforward, highlighting the need for further investigation into the interaction between prediction models and scheduling decisions.

Acknowledgments This research was partially supported by KU Leuven C24E/23/012 'Human-centred decision support based on new theory for personnel rostering'.

References

1. Cardoen, B., Demeulemeester, E., Beliën, J.: Operating room planning and scheduling: A literature review. European journal of operational research **201**(3), 921–932 (2010)
2. Doneda, M., Smet, P., Carello, G., Lanzarone, E., Vanden Berghe, G.: Robust personnel rostering: how accurate should absenteeism predictions be? Journal of Scheduling pp. 1–16 (2025)
3. Lawrance, N., Petrides, G., Guerry, M.A.: Predicting employee absenteeism for cost effective interventions. Decision Support Systems **147**, 113539 (2021)
4. Mandi, J., Kotary, J., Berden, S., Mulamba, M., Bucarey, V., Guns, T., Fioretto, F.: Decision-focused learning: Foundations, state of the art, benchmark and future opportunities. Journal of Artificial Intelligence Research **80**, 1623–1701 (2024)
5. Park, R.T., Stucky, C.H., Moser, C.H.: Machine learning surgery duration predictions compared to traditional methods: A systematic review. Perioperative Care and Operating Room Management **41**, 100581 (2025)
6. Smet, P., Doneda, M., Lanzarone, E., Carello, G.: Prediction accuracy versus rescheduling flexibility in elective surgery management. arXiv preprint arXiv:2507.15566 (2025)
7. Wang, L., Demeulemeester, E., Vansteenkiste, N., Rademakers, F.E.: Operating room planning and scheduling for outpatients and inpatients: A review and future research. Operations Research for Health Care **31**, 100323 (2021)

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Decomposition Techniques for the Aircraft Maintenance Routing Problem

Ida Gjergji[1][0000−0002−2970−1535], Lucas Kletzander[1][0000−0002−2100−7733], Hendrik Bierlee[2][0000−0001−6766−5435], Nysret Musliu[1][0000−0002−3992−8637], and Peter J. Stuckey[3,4][0000−0003−2186−0459]

[1] DBAI, TU Wien, Vienna, Austria
{ida.gjergji,lucas.kletzander,nysret.musliu}@tuwien.ac.at
[2] KU Leuven, Belgium
henk.bierlee@kuleuven.be
[3] Monash University, Melbourne, Australia
[4] OPTIMA ARC Industry Training and Transformation Centre, Melbourne, Australia
peter.stuckey@monash.edu

The aircraft maintenance routing problem (AMRP) is one of the main optimization problems emerging in the airline industry. The AMRP deals with a set of flight legs and a set of aircraft available to operate these flights. In this case, the start and the end times of the flight legs are already known. The aircraft are required to carry out these flight legs together with certain maintenance tasks. The scheduling of the maintenance tasks has to be done in accordance with regulations imposed on the aircraft. A comprehensive overview of work in this area can be found in [3], [8], and [9].

A new version of this problem, AMRP-D (AMRP with a maintenance distribution objective), was introduced by Kletzander et al. [6]. The AMRP-D stems from a real-world application, and it focuses on the even distribution of the maintenance tasks. This objective enables the possibility of aircraft schedules with better utilization of maintenance resources, avoiding both over- and under-utilization, which in the traditional AMRP are disregarded.

Problem Definition The AMRP-D is specified by a set of flight legs, regulations for maintenance, and a set of aircraft including their history assignments. All times are given as integers.

Flight Legs A set of n flight legs $T = \{t_1, \ldots, t_n\}$ is given, each leg t_i is associated with a start time s_i, an end time e_i, and a flight time f_i. Flight legs are not allowed to overlap for the same aircraft.

Maintenance A maintenance on an aircraft cannot overlap with its other maintenance or flight legs, the following types are required:

– Regular and weekly: An aircraft can fly for at most 47 (156) hours after the end of the previous regular (weekly) maintenance, then it needs regular (weekly) maintenance taking 2.5 (7) hours before taking off again. Weekly maintenance includes regular maintenance.

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

– Major (4 types): After at most 950 hours of cumulative flight time since the last maintenance of the same type, the aircraft needs major maintenance taking 14 hours before taking off again. Some types require a single hangar.

Aircraft A set of m aircraft $A = \{a_1, \ldots, a_m\}$ is given including history of maintenance.

Objective The objective of the AMRP-D is to minimize $\sum_s m_s^2$, where m_s shows the total number of aircraft in any type of maintenance for each minute s in the planning period. A second requirement is to schedule the major maintenance tasks as late as possible, without violating feasibility.

Solution Methods The first solution methods for the AMRP-D included a Constraint Programming (CP) model and a decomposition approach [4]. The decomposition approach considers slices of the scheduling period by allocating each flight leg based on its time of occurrence to a single slice. Then the CP model is used in every slice for the assignment of the flight legs to the aircraft. We also presented a Simulated Annealing (SA) approach and a Large Neighborhood Search (LNS) to optimize the objective function of the AMRP-D [4].

Since obtaining a feasible solution is hard for this problem, we previously proposed an alternative CP model targeting the provision of feasible solutions [5]. In [4], the assignment of flight legs to aircraft and the start of maintenance tasks for all aircraft were covered with optional variables. In [5], we investigated if the performance of the model can be enhanced when consecutive flight legs are explicitly modeled, expressing the decision variables for the assignment of flight legs to the aircraft with the regular variables *tail*, which represent the aircraft operating the flight legs. For the start of maintenance tasks that are not necessary in the scheduling period we use dummy tasks. In this work, we present a new integration of this model in the decomposition approach [4].

Additionally, we investigate the decomposition of the AMRP-D in the context of a new column generation (CG) approach [1], [2]. In this decomposition, a potential flight schedule for a single aircraft constitutes a column, while the master problem aims to select a subset of columns that cover each flight leg exactly once. This is known as the set partitioning problem, and can be solved fairly efficiently with modern MIP solvers like Gurobi. On the other hand, composing flight schedules for individual aircraft is much easier than finding a feasible schedule for all aircraft at once. Therefore, this approach has large potential to both find feasible solutions to hard instances, and to optimize the distribution objective in future work.

However, there are also several challenges in the application of column generation to the AMRP-D. First, the hangar non-overlap constraint needs to be modeled in the master problem. This is currently done by counting and restricting the hangar use for each time step, which, depending on time granularity, can significantly increase the complexity of the master problem. A similar approach is required to model the optimization of the maintenance distribution. A further

challenge comes from the fact that typically column generation is started from a feasible solution, but in the case of the AMRP-D, many instances can not be solved to feasibility with simple heuristics. Instead, an infeasible dummy solution with high cost can be utilized, or the coverage of all legs can be seen as an objective rather than a hard constraint.

Preliminary results show that despite the added constraints for hangar coverage, the master problem can be solved efficiently. New columns are currently obtained by a modified version of our CP models, nevertheless, this approach constitutes the current bottleneck and needs to be replaced by a faster method. A custom label setting approach like in related work on bus driver scheduling [7] shows potential to replace the constraint model. Finally, starting from the dummy solution currently shows to be an issue if dummy columns are selected in the integer version of the master problem, which might require branching in a Branch-and-Price approach, or other mitigation strategies.

Acknowledgements This research was funded by the Austrian Science Fund (FWF): 10.55776/I5443 and 10.55776/PIN4864224. The work of P.J. Stuckey is partially supported by the Australian Research Council (OPTIMA ITTC IC200100009).
An earlier version of this work was presented at the KEPS 2025 workshop.

References

1. Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W., Vance, P.H.: Branch-and-price: Column generation for solving huge integer programs. Operations research **46**(3), 316–329 (1998)
2. Desrosiers, J., Lübbecke, M., Desaulniers, G., Gauthier, J.B.: Branch-and-price. Springer (2024)
3. Eltoukhy, A.E., Chan, F.T., Chung, S.H.: Airline schedule planning: a review and future directions. Industrial Management & Data Systems **117**(6), 1201–1243 (2017)
4. Gjergji, I., Kletzander, L., Bierlee, H., Musliu, N., Stuckey, P.: Combining constraint programming and metaheuristics for aircraft maintenance routing with a distribution objective. In: Tack, G. (ed.) Integration of Constraint Programming, Artificial Intelligence, and Operations Research. pp. 1–17. Springer Nature Switzerland, Cham (2025)
5. Gjergji, I., Kletzander, L., Bierlee, H., Musliu, N., Stuckey, P.: A consecutive flight leg model for the aircraft maintenance routing problem. KEPS workshop (2025)
6. Kletzander, L., Gjergji, I., Musliu, N.: Combining aircraft maintenance routing with a distribution objective. In: Proceedings of the 14th International Conference on the Practice and Theory of Automated Timetabling, PATAT 2024. pp. 325–328 (2024)
7. Kletzander, L., Musliu, N., Van Hentenryck, P.: Branch and price for bus driver scheduling with complex break constraints. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 11853–11861 (2021)
8. Temucin, T., Tuzkaya, G., Vayvay, O.: Aircraft maintenance routing problem–a literature survey. Promet-Traffic&Transportation **33**(4), 491–503 (2021)
9. Xu, Y., Wandelt, S., Sun, X.: Airline scheduling optimization: Literature review and a discussion of modeling methodologies. Intelligent Transportation Infrastructure (2023)

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

KHE26: Progress towards an Efficient, Practical Nurse Rostering Solver (Abstract)

Jeffrey H. Kingston

School of Information Technologies, The University of Sydney, Australia
jeff@it.usyd.edu.au
http://jeffreykingston.id.au

Abstract. This paper presents ongoing work on a solver for the nurse rostering problem which is oriented towards practice, in that it aims to find good solutions quickly for a wide range of instances. This abstract shows that the current version of the solver, KHE26, has improved significantly on the previous version on one important data set, of four-week instances from the Second International Nurse Rostering Competition. Future work will present results for this as well as several other data sets, and will explain how these improvements have been obtained without fundamental changes to the solver.

Keywords: Automated timetabling · Nurse rostering

1 Introduction

Nurse rostering, the problem of assigning nurses to shifts in hospital wards, is one of the most-studied problems in automated timetabling.

In a previous paper [7], this author presented KHE24, an open source solver [5] for nurse rostering. KHE24 reads and writes instances and solutions in the XESTT data format [6], in which all of the well-known nurse rostering data sets have been expressed, plus solutions from various sources which are among the best known.

This paper presents KHE26, the current version of that solver. Rather than trying to find new best solutions, KHE24 and KHE26 try to find solutions whose cost is within 10% of the best known, given a 5 minute run on ordinary desktop hardware, and within 5% of the best known, given a 60 minute run on ordinary desktop hardware. If this can be done for a wide range of real-world instances, then the solver will be more useful in practice than solvers that are targeted to specific kinds of instances, or that run for many hours.

The work reported by the previous paper was avowedly incomplete. KHE24 performed well on two data sets: the Curtois original instances [3] and the First International Nurse Rostering Competition [4] instances. But it performed poorly on a standard set of four-week instances from the Second International Nurse Rostering Competition [1, 2]. Table 4 from [7] showed that, given 5 minutes per instance, KHE24 produced solutions for those instances which were on average 34% worse than the best known solutions.

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

KHE26, the version of the solver presented here, is much improved. Table 1 presents KHE26's results for the same set of four-week instances from the Second International Nurse Rostering Competition. It shows that, given 5 minutes per instance, KHE26 is producing solutions which are on average 10% worse than the best known solutions. While this is not mission accomplished, because 10% on average is not the same as 10% for each instance, it is a significant improvement on 34% and is approaching the desired outcome.

In addition to presenting results on other data sets, future work will explain how this improvement was obtained from a solver which, in outline at least, has not greatly changed. But, in brief, it is the result of detailed analysis of the solutions found by the solver, aiming to identify weaknesses in the algorithm. As these are corrected, small improvements in solution quality accumulate.

References

1. Sara Ceschia, Nguyen Thi Thanh Dang, Patrick De Causmaecker, Stefaan Haspeslagh, and Andrea Schaerf, Second international nurse rostering competition (INRC-II), problem description and rules. oRR abs/1501.04177 (2015). http://arxiv.org/abs/1501.04177
2. Sara Ceschia, Nguyen Thi Thanh Dang, Patrick De Causmaecker, Stephaan Haspeslagh, and Andrea Schaerf, Second international nurse rostering competition (INRC-II) web site, URL http://mobiz.vives.be/inrc2/.
3. Tim Curtois, Employee Shift Scheduling Benchmark Data Sets, http://www.schedulingbenchmarks.org/ (2019)
4. Stefaan Haspeslagh, Patrick De Causmaecker, Martin Stølevik, and Andrea Schaerf, First international nurse rostering competition website, URL: http://www.kuleuven-kortrijk.be/nrpcompetition (2010)
5. Jeffrey H. Kingston, KHE web site, http://jeffreykingston.id.au/khe (2014).
6. Jeffrey H. Kingston, XESTT web site, http://jeffreykingston.id.au/xestt (2017)
7. Jeffrey H. Kingston, KHE24: Towards a Practical Solver for Nurse Rostering (Abstract), PATAT 2024 (Fourteenth International Conference on Practice and Theory of Automated Timetabling, Copenhagen, Denmark, August 2024).
8. A. Legrain, J. Omer, and S. Rosat, A rotation-based branch-and-price approach for the nurse scheduling problem (2017). Working paper, available at https://hal.archives-ouvertes.fr/hal-01545421.

Table 1. KHE26 results on 20 standard four-week instances from the Second International Nurse Rostering Competition [1, 2], obtained on 26 February 2026. For each instance, column LOR17 shows the cost of the best known solution, as reported in [8] and obtained from the authors of that paper by this author. The running time is reported in [8] to conform to the competition rules but is not present in the file obtained from its authors, so does not appear here; it is about 20 minutes for the largest instances. Column KHE26x12 shows the cost of the best solution obtained by KHE26 over 12 simultaneous 5-minute runs, each with a different random seed, on the author's 12-core desktop computer (the same computer as the one used for testing in the author's KHE24 paper); the ratio of that cost to the cost of the best known solution; and the running time in seconds of the whole run of 12 simultaneous solves (that is, the wall clock time). Choosing the best of several solutions is an integral part of the KHE26 approach, not something that is done just for testing. It is a simple way to exploit the power of multi-core computers, which are now normal on desktops. The XESTT archive file from which this table is derived may be found online, at `http://jeffreykingston.id.au/tt_solns/KHE26-2026-02-26-INRC2-4.xml.gz`. This file contains the actual solutions, not just their costs.

Instances (20)	**LOR17**			**KHE26x12**		
	Cost	**Rel.**	**Time**	**Cost**	**Rel.**	**Time**
INRC2-4-030-1-6291	**1695**	1.00	-	1780	1.05	283.8
INRC2-4-030-1-6753	**1890**	1.00	-	1960	1.04	294.4
INRC2-4-035-0-1718	**1425**	1.00	-	1525	1.07	300.1
INRC2-4-035-2-8875	**1155**	1.00	-	1270	1.10	285.6
INRC2-4-040-0-2061	**1685**	1.00	-	1845	1.09	300.2
INRC2-4-040-2-6106	**1890**	1.00	-	2030	1.07	300.1
INRC2-4-050-0-0487	**1505**	1.00	-	1615	1.07	300.4
INRC2-4-050-0-7272	**1500**	1.00	-	1520	1.01	300.2
INRC2-4-060-1-6115	**2505**	1.00	-	2830	1.13	300.4
INRC2-4-060-1-9638	**2750**	1.00	-	3105	1.13	300.2
INRC2-4-070-0-3651	**2435**	1.00	-	2700	1.11	300.5
INRC2-4-070-0-4967	**2175**	1.00	-	2475	1.14	300.5
INRC2-4-080-2-4333	**3340**	1.00	-	3665	1.10	300.6
INRC2-4-080-2-6048	**3260**	1.00	-	3660	1.12	300.4
INRC2-4-100-0-1108	**1245**	1.00	-	1410	1.13	300.5
INRC2-4-100-2-0646	**1950**	1.00	-	2230	1.14	300.4
INRC2-4-110-0-1428	**2440**	1.00	-	2640	1.08	300.3
INRC2-4-110-0-1935	**2560**	1.00	-	2820	1.10	300.4
INRC2-4-120-1-4626	**2170**	1.00	-	2525	1.16	300.3
INRC2-4-120-1-5698	**2220**	1.00	-	2575	1.16	300.3
Average	**2090**	1.00		2309	1.10	298.5

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Educational Timetabling with Unconstrained Demand Forecasting

Olivia Phillips[0009−0006−9886−5492], Dima Nazzal[0000−0003−1137−2603], and Joel Sokol[0000−0001−8228−8359]

Georgia Institute of Technology, Atlanta GA 30332, USA
ogphillips@gatech.edu

Abstract. The University Course Timetabling Problem (UCTTP) often assumes that course enrollment is given a priori, though this is not always the case in practice. In this extended abstract, we outline our work to predict unconstrained course demand and student registration behavior using methods borrowed from airline revenue management, and to analyze the impact of these predictions within a timetabling optimization model.

Keywords: Educational timetabling · Demand forecasting · Overbooking.

1 Introduction

In the University Course Timetabling Problem (UCTTP), a set of courses is assigned to times, classrooms, students, and instructors. This problem has been studied extensively, with many variations and solution methods (see surveys [3, 4, 6]). Although most formulations of the problem are NP-complete [2], there exist methods for finding high-quality solutions to large-scale instances in practice [8].

In most of these studies, course enrollment is assumed to be fixed in order to meet classroom capacity constraints. In practice, universities must often estimate course enrollment when making room assignments without knowing the true course demand or how many students will add or drop the course. One such institution is the Georgia Institute of Technology (GT), the focus of our case study. At GT, the registrar assigns centrally-scheduled courses to classrooms before the first phase of registration opens to students. Most enrollment predictions are rolled over from the previous semester each course was offered; when this is not possible, the academic unit supplies its best guess. The registrar's goals are to use these numbers to maximize classroom utilization while satisfying the building preferences of each academic unit.

Given that this method has been in place for years, it provides a reasonable estimate for enrollment. However, this method does not account for constrained demand. If a course was assigned a room with a capacity of 30 last semester, and then enrolled 30 students, the final enrollment would not include the demand of students who wished to take the course but could not because of the

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

limited capacity. Most work investigating unconstrained demand for timetabling has utilized student surveys to estimate course demand [9, 11], which doesn't capture students' actual registration behavior. Additionally, we have only found one paper that defines a mathematical model for the add/drop process [13]. If we knew how many students would enroll in a course during registration, we could assign classrooms while satisfying capacity constraints; and if we knew students' add/drop behavior, we could allow for overbooked classrooms.

Although largely unstudied in the context of education, demand forecasting and overbooking have been key areas of research in the airline industry for decades. In [10], we find a taxonomy of the various methods of demand unconstraining used in the literature over the years, bucketed into basic approaches (ignoring censored data, imputation, etc.), statistical methods (time series, linear regression, etc.), discrete choice models, and optimization methods (expectation maximization, nonlinear programming, etc.). The above survey also provides examples of how these techniques for unconstraining demand have been adopted by other industries, including rail, hotel, rental, and retail; we propose adding university operations to that list.

Student registration behavior over time also parallels airline passenger behavior. Students often over-enroll in courses and later drop them, thus artificially inflating the demand for a course. In a similar way, airline passengers often purchase tickets and then cancel their flight at the last minute or no-show altogether. With average no-show rates of 10% during regular operations and 20% during the peak holiday season, not overbooking means that these seats would remain empty when they could otherwise be filled with paying customers [1]. Similarly to demand forecasting, overbooking has been studied extensively in the context of airline operations [5], but the only applications of overbooking in educational institutions we have found study the institution-level yield rates, not course-level enrollments [12].

In order to make better use of university resources, we propose adapting demand unconstraining methods and overbooking models from the airline industry to better predict student enrollment and registration behavior and thus improve the inputs to our timetabling model. Our work therefore bridges the gap between stochastic enrollment behavior and traditionally deterministic course timetabling.

2 Methods

2.1 Initial Formulation

We base our model on the formulation in [7]. In the first stage of our analysis, we assume a fixed timetable and assign classrooms to maximize classroom utilization. We have a set of class sections $\mathcal{X}$, each with projected enrollment p_x. For every day-time pair in the set of start times $\mathcal{ST}$, we define $\mathcal{X}_{d,t}$, the set of sections that have a meeting on day d that starts at or before time t on that day and makes the classroom it is assigned to unavailable at that time. We also have

a set of classrooms $\mathcal{R}$, each with capacity n_r. To allow for overbooking, we let M be the maximum percentage by which classrooms may be overfilled, and β be the factor by which overbooked seats are penalized compared to underbooked seats.

We define the following decision variables.

$$X_{x,r} = \begin{cases} 1 & \text{if section } x \text{ is assigned to room } r \\ 0 & \text{otherwise} \end{cases}$$

$$Y_x \in [0, M]: \quad \text{percent by which section } x \text{ is overbooked}$$

The primary objective is to maximize classroom utilization, which is equivalent to minimizing unused capacity. We add a penalty term proportional to the number of overbooked seats.

$$\min \sum_{x \in \mathcal{X}} \sum_{r \in \mathcal{R}} \max(0, n_r - p_x) X_{x,r} + \beta \sum_{x \in \mathcal{X}} p_x Y_x \tag{1}$$

Our constraints ensure that each room is assigned to at most one class section at a given time, and each class section is assigned to a room. We relax the room capacity constraints by adding a term to allow for controlled overbooking.

$$\sum_{x \in \mathcal{X}_{d,t}} X_{x,r} \leq 1 \qquad \forall r \in \mathcal{R},\ \forall (d,t) \in \mathcal{ST} \tag{2}$$

$$\sum_{r \in \mathcal{R}} X_{x,r} = 1 \qquad \forall x \in \mathcal{X} \tag{3}$$

$$p_x X_{x,r} \leq n_r + p_x Y_x \qquad \forall x \in \mathcal{X},\ \forall r \in \mathcal{R} \tag{4}$$

The initial dataset that we used to test the above formulation consists of approximately 2,000 course sections, 150 classrooms, and 125,000 total projected enrollments across all sections for a given semester at Georgia Tech.

2.2 Predicting Demand

While overbooking can allow for an improved objective value, overbooking too aggressively can incur costs that outweigh the benefits. While the airline industry can reroute passengers and accept the accompanying denied boarding cost, universities cannot force a student to unregister from a course due to capacity issues. They thus have two options: move the course to a larger room or change the course modality. The first is challenging and potentially not feasible, since the problem is already highly constrained. On the other hand, changing modality may incur technology costs, as well as frustration from students and faculty who were expecting an in-person course. We thus seek to control these over-enrollments in our model.

To conduct these analyses, we are acquiring student registration data from the last three years at GT, including add/drop behavior and the precise timing

of these actions. In order to unconstrain demand for courses at capacity, we will use time series and discrete choice models to account for both the temporal and strategic nature of student registration behavior when considering alternatives throughout the registration period. Additionally, we will create models to predict the future enrollments of a given student based on factors like their course history and program of study. Finally, we will analyze patterns of student registration behavior over time to predict how many students will drop a course and how many students admitted from the waitlist will actually end up enrolling in the course. These predictions will allow us to recommend an overbooking level for each course, which we will then use in the above optimization formulation with relaxed capacity constraints. In a later step, we will drop the assumption that class times are fixed, and analyze the impact of better demand prediction on the overall university timetable. To our knowledge, this is the first work to incorporate demand unconstraining and overbooking methods into course-level educational timetabling.

References

1. Belobaba, P.: Airline Revenue Management. In: The Global Airline Industry. John Wiley & Sons, Incorporated, United Kingdom (2015)
2. Carter, M.W., Tovey, C.A.: When Is the Classroom Assignment Problem Hard? Operations Research **40**, S28–S39 (1992). https://doi.org/10.1287/opre.40.1.S28
3. Ceschia, S., Di Gaspero, L., Schaerf, A.: Educational timetabling: Problems, benchmarks, and state-of-the-art results. European Journal of Operational Research **308**(1), 1–18 (Jul 2023). https://doi.org/10.1016/j.ejor.2022.07.011
4. Chen, M.C., Sze, S.N., Goh, S.L., Sabar, N.R., Kendall, G.: A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities. IEEE Access **9**, 106515–106529 (2021). https://doi.org/10.1109/ACCESS.2021.3100613
5. McGill, J., van Ryzin, G.: Revenue Management: Research Overview and Prospects. Transportation Science **33**, 233–256 (May 1999). https://doi.org/10.1287/trsc.33.2.233
6. MirHassani, S.A., Habibi, F.: Solution approaches to the course timetabling problem. Artificial Intelligence Review **39**(2), 133–149 (Feb 2013). https://doi.org/10.1007/s10462-011-9262-6
7. Navabi-Shirazi, M., El Tonbari, M., Boland, N., Nazzal, D., Steimle, L.N.: Multicriteria Course Mode Selection and Classroom Assignment Under Sudden Space Scarcity. Manufacturing & Service Operations Management **24**(6), 3252–3268 (Nov 2022). https://doi.org/10.1287/msom.2022.1131
8. Phillips, A.E., Waterer, H., Ehrgott, M., Ryan, D.M.: Integer programming methods for large-scale practical classroom assignment problems. Computers & Operations Research **53**, 42–53 (Jan 2015). https://doi.org/10.1016/j.cor.2014.07.012
9. Sampson, S.E., Freeland, J.R., Weiss, E.N.: Class Scheduling to Maximize Participant Satisfaction. Interfaces **25**(3), 30–41 (Jun 1995). https://doi.org/10.1287/inte.25.3.30
10. Sharif Azadeh, S., Marcotte, P., Savard, G.: A taxonomy of demand uncensoring methods in revenue management. Journal of Revenue & Pricing Management (Apr 2014). https://doi.org/10.1057/rpm.2014.8

11. Thompson, G.M.: Using information on unconstrained student demand to improve university course schedules. Journal of Operations Management **23**(2), 197–208 (Feb 2005). https://doi.org/10.1016/j.jom.2004.07.006
12. Zhang, L., Pippins, J.C.: Application of overbooking model to college admission at The Citadel. International Transactions in Operational Research **28**(5), 2402–2413 (2021). https://doi.org/10.1111/itor.12948
13. Özer, A.H., Özturan, C.: A direct barter model for course add/drop process. Discrete Applied Mathematics **159**(8), 812–825 (Apr 2011). https://doi.org/10.1016/j.dam.2011.01.004

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Novel neighborhood structures for incomplete round robin tournaments

Karel Devriesere[1 [0000−0002−6774−849X]], David Van Bulck[1 [0000−0002−1222−4541]], and Dries Goossens[1 [0000−0003−0224−3412]]

Ghent University, Department of Business Administration and Operations Management, Belgium. david.vanbulck@ugent.be, karel.devriesere@ugent.be, dries.goossens@ugent.be

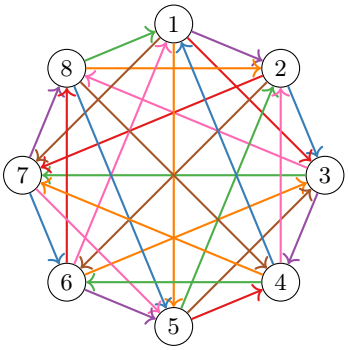
1 Sports timetabling as a graph problem

In this work, we study novel neighborhood structures for incomplete round robin (iRR) tournaments. In this tournament format, we are given a set of n teams and a set of r rounds, such that $r < n - 1$, i.e. not every team can face every other team. Hence, in contrast to round robin tournaments (RR), we have to determine the subset of games that is played. In graph theory terms, if we let teams correspond to vertices and matches correspond to edges, a conventional round robin tournament can be seen as a proper edge coloring of the complete graph K_n, where the color denotes the round in which a match is played. In contrast, in an incomplete round robin tournament, some of the edges are uncolored (meaning that these matches are not scheduled), and we have to find an r-regular subgraph of K_n that is properly edge colored with r colors, while all other edges in K_n remain uncolored. In this case, we refer to the coloring as a *partial proper edge coloring*. This difference is illustrated in Figures 1 and 2 for a RR and iRR tournament with 8 teams, respectively, where only 5 rounds are played in the iRR tournament.

2 Related literature

The iRR format has gained significantly in popularity since its introduction in the three major European football competitions, i.e. the UEFA Champions, Europa and Conference league, respectively [1, 3, 4, 8]. Moreover, it has also seen its introduction in youth sports competitions. In particular, it has been shown that the iRR format significantly reduces travel time compared to the format in which teams are partitioned into leagues and each league is organized as a double round robin tournament [2, 7]. Despite this gain in popularity, literature about neighborhood structures for iRR tournaments is sparse. Indeed, existing neighborhood structures for sports scheduling have so far been developed almost solely for round robin tournaments. Although their nomenclature differs throughout the literature, they are commonly known as Team Swap, Partial Team Swap, Round Swap, Partial Round Swap, and Cycle-and Path reversal [5, 6]. As a first contribution, we show that these neighborhood structures remain applicable for iRR tournaments.

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

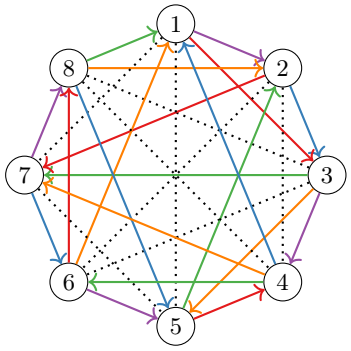


(a) Proper edge coloring of K_8

R1	R2	R3	R4	R5	R6	R7
3-1	3-2	7-3	4-3	3-6	8-3	3-5
7-2	1-4	2-5	2-1	2-8	2-4	6-2
8-6	5-8	1-8	8-7	5-1	1-6	7-1
4-5	6-7	6-4	5-6	7-4	5-7	4-8

(b) Corresponding RR timetable to the colored graph in a)

Fig. 1: Relation between proper edge coloring and RR timetable



(a) Partial proper edge coloring with 5 colors

R1	R2	R3	R4	R5
3-1	3-2	7-3	4-3	1-6
7-2	1-4	2-5	2-1	2-8
8-6	5-8	1-8	8-7	5-3
4-5	6-7	6-4	5-6	7-4

(b) Corresponding iRR timetable with 5 rounds to the colored graph in a)

Fig. 2: Relation between partial proper edge coloring and iRR timetable

3 Contribution

A major shortcoming of these neighborhood structures, however, is their inability to reduce non-isomorphic matches into the timetable, meaning that they never structurally modify the timetable. We fill this gap by proposing two different neighborhood structures: *incomplete Partial Team Swap* (iPTS) and *incomplete Partial Round Swap* (iPRS). iPTS swaps the opponents of two teams over a subset of the rounds, such that at most one uncolored edge is included in the swap for each of the two teams. iPRS takes as input a cycle alternating between uncolored edges and edges colored with one particular color, and swaps the colored for the uncolored edges. Hence, both iPTS and iPRS introduce new games into the iRR timetable. In fact, we show that our neighborhood structures are able to

introduce non-isomorphic matches into the timetable, and prove solution space connectivity for one of them in case $r \leq \frac{n}{2}$. Moreover, we embed the neighborhood structures into a late acceptance hill climbing algorithm (LAHC) and test the algorithm on three sets of instances from the literature. Table 1 shows preliminary results for one of the considered instance sets, stemming from Belgian youth hockey competitions (see [2]). The LAHC with the proposed neighborhood structures is here compared to solving a straightforward integer program (IP) and using a metaheuristic proposed by [7] (GM), which is based on solving a sequence of greedy matchings and is the state of the art algorithm to construct iRR timetables. A time limit of 2 hours is used. The LAHC is seen to outperform the other two methods and produces solutions close to the lower bound (LB), thereby empirically confirming the validity of the proposed neighborhood structures.

Table 1: Preliminary results for some selected instances

				GM		LAHC	
Instance	LB	IP-UB	Gap (%)	UB	Gap (%)	UB	Gap (%)
i01	2821	**2821**	0.0	*2903*	2.82	**2821**	0.0
i02	3980	**3980**	0.0	*4025*	1.12	**3980**	0.0
i03	150381	*289065*	47.98	158058	4.86	**156747**	4.06
i04	180310	*339237*	46.85	190861	5.53	**186152**	3.14
i05	220520	*391507*	43.67	231584	4.78	**227254**	2.96
i06	175592	*301517*	41.76	187216	6.21	**181371**	3.19

The best found upper bound (UB) is marked in bold, while the worst found upper bound is italicized.

Acknowledgements David Van Bulck thanks the support of Research Foundation Flanders (FWO) [23AXE35N].

References

1. Csató, L., Ilyin, S.: Misaligned incentives in sports: a mathematical analysis of the post-2024 UEFA Champions League qualification. IMA Journal of Management Mathematics **36**(3), 579–591 (2025)
2. Devriesere, K., Goossens, D.: Redesigning Belgian youth field hockey competitions using an incomplete round-robin tournament. INFORMS Journal on Applied Analytics **55**(6), 437–505 (2025)
3. Guyon, J., Meunier, F., Ben Salem, A., Buchholtzer, T., Tanré, M.: Drawing and scheduling the UEFA Champions League league phase (2025), available at SSRN: https://ssrn.com/abstract=5413142
4. Gyimesi, A.: Competitive balance in the post-2024 Champions League and the European Super League: A simulation study. Journal of Sports Economics **25**(6), 707–734 (2024)

5. Januario, T., Urrutia, S., de Werra, D.: Sports scheduling search space connectivity: A riffle shuffle driven approach. Discrete Applied Mathematics **211**, 113–120 (2016)
6. Knust, S., von Thaden, M.: Balanced home–away assignments. Discrete Optimization **3**(4), 354–365 (2006)
7. Li, M., Van Bulck, D., Goossens, D.: Beyond leagues: A single incomplete round robin tournament for multi-league sports timetabling. European Journal of Operational Research **323**(1), 208–223 (2025)
8. Winkelmann, D., Michels, R., Deutscher, C.: Predicting Qualification Thresholds in UEFA's Incomplete Round-Robin Tournaments. Manuscript. URL: https://arxiv.org/abs/2508.20075 (2025)

Case Study: An Integrated Airport Security Planning Approach Using Simulated Annealing and Minimum Cost Flow

Hicham Kamhi and Lotte Berghman

ORTEC B.V., Zoetermeer EA 2719, The Netherlands

Abstract. This research develops an integrated optimization framework for monthly rostering of airport ground security personnel. The approach combines a Simulated Annealing metaheuristic with a Minimum Cost Flow based staffing assessment to efficiently evaluate coverage and guide the search. Using a real-world dataset from airport security operations, the framework addresses the practical complexity of simultaneously constructing shifts and assigning them to employees while satisfying labor regulations, contractual constraints and employee preferences.
Initial rosters are generated through greedy construction heuristics and subsequently improved using neighborhood moves that modify assignments across multi-day blocks and refine shift characteristics such as timing, duration, and break placement. The Minimum Cost Flow formulation matches available employees to skill demand in each period, enabling fast evaluation of under- and overstaffing across the planning horizon.
For the dataset of the practical case study, the resulting rosters achieve staffing coverage above 99% with less than 1% overstaffing. Overtime remains controlled: roughly 3% of employees work overtime, with average overtime per working employee below 0.5 hours. External staffing is limited to around 6%. Sensitivity analyses illustrate how rosters change when different KPIs are prioritized.

Keywords: Employee Scheduling · Case Study · Simulated Annealing

1 Introduction

The airport security industry benefits greatly from automated and optimized scheduling. This research presents a case in which ORTEC, a leading provider of workforce scheduling optimization software, developed an approach to improve the monthly rostering process for airport ground security personnel. The aim was to design a framework that efficiently allocates staff resources while accounting for the operational complexities inherent to airport security work.

Airport security consists of predefined activities that must be staffed throughout the day, with personnel demand specified per half-hour period and known three months in advance. These activities vary in required staffing levels and can only be performed by employees who hold the appropriate skills. Once the

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

monthly staffing requirements are known, the security company must design shifts and assign them to employees while balancing multiple objectives and satisfying multiple constraints.

In long-term planning, the assignment of employees to specific activities within their shifts is not yet considered. Instead, the objective is to create shift configurations and shift demands that cover the total staffing demand for each skill in every half-hour period. These requirements are obtained by aggregating the demands of all activities that require the same skill. A roster that meets these aggregated skill demands will, by extension, cover the underlying activities. To accomplish this, the rostering problem involves determining feasible shift configurations, each starting at any half-hour and lasting between 4 and 9.5 hours, and assigning them to employees. Because employees may possess multiple skills, they can cover different skill requirements within the same shift, though only one skill can be covered at a time. This flexibility enables closer alignment between staffing levels and fluctuating demand.

The roster must satisfy several types of constraints. First, all schedules must comply with labor rules based on the Dutch Working Hours Act (ATW). These include requirements on daily and weekly rest, restrictions on Sunday and night work, limitations on consecutive night shifts, and mandatory breaks for longer shifts. All employees must satisfy these rules for the roster to be feasible. Second, contractual requirements specify total weekly hours and allowable shift lengths. Across a month, an employee may not exceed 120% of their contracted hours, with the excess counted as overtime. Each assigned shift must lie within the contract's minimum and maximum lengths. Finally, employee preferences are modeled as soft constraints. Employees may request days on or off and may indicate preferred lengths of consecutive shift sequences. While these preferences may be violated, doing so reduces overall solution quality.

Data Case Study The case study covers a 28-day planning horizon and a workforce of 657 employees, of whom 487 are internal. Employees hold an average of 2.3 skills, drawn from a set of 7 unique skills. Total skill demand over the horizon equals 22,510 working hours. Figure 1 shows the mean demand per half-hour period, calculated across all 28 days for each skill. Demand starts at period 8 (04:00–04:30) and falls to zero after period 43 (21:30–22:00), fluctuating throughout the day with morning and evening peaks and a minor bump early in the afternoon.

2 Methodology

This research addresses the integrated monthly rostering problem in which shifts must be constructed and assigned to employees while satisfying labor regulations, contractual constraints, and employee preferences. We solve this problem using a Simulated Annealing (SA) metaheuristic, where solution quality is evaluated through a staffing assessment step that matches scheduled employees to skill demand using a Minimum Cost Flow (MCF) formulation.

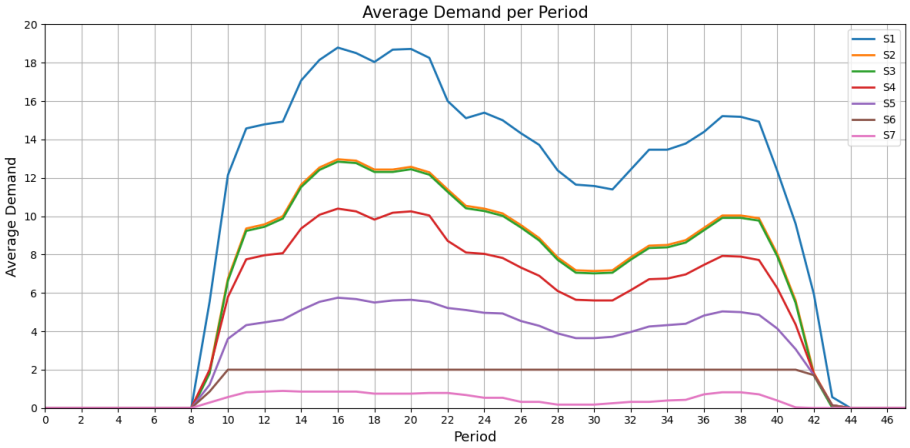


Fig. 1: Mean demand per half-hour period (periods 0–47) for each skill, averaged across all 28 days of the planning horizon.

The solution method is integrated in the sense that it searches directly in the space of complete rosters rather than separating shift generation and shift assignment into sequential steps. The search is initialized with a feasible roster produced by a greedy construction procedure. In this construction phase, employees are scheduled one-by-one using a fixed sorting order that prioritizes internal employees and preserves flexibility by keeping scarce and multi-skilled employees available for later decisions. For each employee, the heuristic assigns shifts only when they cover unmet demand and remain feasible with respect to all hard constraints; for longer shifts, the break position is chosen such that it yields the lowest penalty.

SA then improves this initial roster using two complementary move classes. Assignment searchers change who works which shifts: they swap x-day blocks between two employees (vertical swap), swap two non-overlapping x-day blocks within a single employee schedule (horizontal swap), or replace an employee's x-day block with a single uniform shift type, including days off (change move). Shift searchers refine existing shifts by moving the break, shifting the entire shift one period earlier or later, or extending/shortening the shift by one period at either the start or the end.

Staffing assessment is recomputed after roster changes by matching available employees to skill demand in each period. By combining demand slots and employees with identical skills, this Assignment Problem can be formulated as a Minimum Cost Flow problem and solved using the Cost-Scaling Push-Relabel algorithm [1], which is in practice substantially faster on sparse graphs than the Hungarian Method [2]. Solving the flow model yields the implied under- and overstaffing per skill and period, which is then used to evaluate roster quality throughout the search.

3 Results

The algorithm delivers high-quality solutions according to the defined key performance indicators. Results show excellent staffing coverage above 99%, with controlled over-staffing of less than 1%. Overtime is managed effectively, with about 3% of employees working overtime and average overtime per working employee under 0.5 hours, while taking into account employee preferences for work schedules and preferred working block lengths. Use of external staff is kept at an appropriate level around 6% to support internal capacity when necessary. Furthermore, we did a sensitivity analysis to compare rosters where one of the indicators like under-staffing, over-staffing, overtime, external staffing, or preferences, is valued higher. This shows that the framework is also interesting for other companies and datasets where prioritization is different.
In conclusion, the framework produces practical, high-quality rosters that meet coverage requirements while balancing operational needs with employee preferences. It offers a scalable and adaptable solution for airport security operations and can be applied to other industries requiring continuous staffing, providing a strong foundation for further improvement and real-world implementation.

References

1. A. V. Goldberg and R. E. Tarjan. Finding minimum-cost circulations by successive approximation. Proceedings of the nineteenth annual ACM symposium on Theory of computing, 15(3):7–18, 1990.
2. H. W. Kuhn. The hungarian method for the assignment problem. Naval Research Logistics Quarterly, 2(1-2):83–97, 1955.

A Hybrid Adaptive Metaheuristic-based Q-Learning Approach for the Home Healthcare Routing and Scheduling Problem

Yun-Chia Liang[1], Angela Hsiang-Ling Chen[2], Sebastian Gunawan[2], and Vanny Minanda[1]

[1] Department of Industrial Engineering and Management, Yuan Ze University, Taoyuan City, Taiwan ROC, 320
[2] Department of Industrial and Systems Engineering, Chung Yuan Christian University, Taoyuan City, Taiwan ROC, 320
ycliang@saturn.yzu.edu.tw

Abstract. This study addresses the Home Healthcare Routing and Scheduling Problem (HHCRSP), a complex extension of the Vehicle Routing Problem involving joint nurse–patient assignment, service scheduling, and route planning under synchronized multi-nurse constraints and strict time windows. To tackle the highly constrained search space, we propose a Hybrid Adaptive Metaheuristic-based Q-Learning (AHMQL). The approach utilizes a rigorous survival-selection mechanism to maintain population diversity while leveraging Q-learning as an adaptive memory scheme to guide local search operations. The proposed algorithm was evaluated on benchmark instances of varying scales (up to 100 patients) representing high-density routing scenarios. The results demonstrate promising scalability and high constraint-fulfillment ratios in medium- and large-scale instances, approaching MILP fitness function with faster CPU Time. Finally, we discuss computational trade-offs, particularly regarding convergence rates in large-scale scenarios, and outline paths for future algorithmic refinement.

Keywords: Home Healthcare Routing and Scheduling Problem · Metaheuristic · Q-learning

1 Introduction

Taiwan's rapid population aging creates intense pressure on healthcare delivery, particularly for rural elderly patients who face severe geographic and infrastructural barriers to conventional medical facilities [1, 2]. While home healthcare has emerged as a vital model to improve patient quality of life and optimize resource efficiency, managing it requires solving the Home Healthcare Routing and Scheduling Problem (HHCRSP) [3, 4]. As an advanced extension of the Vehicle Routing Problem with Time Windows (VRPTW), HHCRSP introduces highly complex, healthcare-specific constraints. These include strict synchronized multi-caregiver visits (where certain treatments require multiple nurses to arrive simul-

taneously), localized time windows, and strict caregiver-qualification matching [5].

The combinatorial complexity of these overlapping operational constraints heavily restricts the feasible solution space, causing some state-of-the-art metaheuristics to frequently stall in local optima. To overcome this limitation, this study develops a sophisticated hybrid framework that couples a population-based metaheuristic with Reinforcement Learning (RL). While RL has seen extensive success in sequential decision-making tasks like robotics, its integration into metaheuristic frameworks for logistical optimization remains underutilized [6, 7].

The fundamental motivation for developing this hybrid approach is twofold: (1) Algorithmic Synergy: Population-based selection mechanics provide robust global exploration capabilities across the discrete solution space, while the adaptive learning memory of Q-learning dynamically tunes neighborhood selection parameters based on search history, mitigating premature convergence; (2) Practical Utility: From a managerial perspective, healthcare providers require an automated scheduling mechanism capable of synchronizing caregiver timetables, minimizing unproductive travel/idle times, and strictly eliminating service-window violations to maximize patient safety and satisfaction. The primary aim of this study is to validate whether embedding adaptive learning memory into a metaheuristic framework can reliably solve high-density, real-world scale routing and scheduling instances where operational constraints are tight.

2 Methodology

This study proposes a Hybrid Adaptive Metaheuristic-based Q-Learning (AHMQL) to solve the highly constrained HHCRSP. Instead of relying on static heuristic parameters, the framework embeds a central Q-table, $Q(d, i, j)$, to act as an adaptive spatial-temporal memory layer that stores the learned expected utility of moving from location i to location j on day d. Over successive iterations, the algorithm maintains a candidate schedule population V, constructing solutions sequentially via an adaptive state transition rule where traditional heuristic visibility is dynamically weighted against historical Q-values. At the end of each iteration, a rank-based survival selection routine filters the population into elite (V_{elite}) and feasible ($V_{feasible}$) tiers based on fitness function of maximizing patient visit fulfillment. High-quality routing trajectories from these surviving schedules are then backpropagated into the central Q-table using temporal-difference learning. This iterative feedback loop structurally guides subsequent iterations toward high-yield routing choices while preserving enough population diversity to prevent the search from stalling in local optima, as formalized in Algorithm 1.

Algorithm 1. Hybrid Adaptive Metaheuristic-based Q-Learning for HHCRSP

1. Initialize $Q(d, i, j) \leftarrow 0 \ \forall d \in D,\ i, j \in P \cup \{0\}$ and set $K^* \leftarrow \varnothing$.

2. For iteration $t = 1$ to T do.
3. Compute current exploration rate:

$$\varepsilon_t = \varepsilon_{start} - \left(\frac{t-1}{T-1}\right)(\varepsilon_{start} - \varepsilon_{end}).$$

4. Initialize an empty candidate solution population $V \leftarrow \varnothing$.
5. For candidate index $k = 1$ to M do.
6. Initialize schedule $K_k \leftarrow \varnothing$, visit counters, and caregiver spatial-temporal states.
7. For each day $d \in D$ do.
8. While feasible service decisions remain do.
9. Select lead nurse $n \in N$ and define $S = (d, i)$.
10. Construct feasible set Ω_k based on demand, qualification, and time window bounds.
11. If $\Omega_k = \varnothing$ then break or transition to alternative nurse.
12. Sample random variable $r \sim U(0, 1)$.
13. If $r < \varepsilon_t$ then select next patient $j \in \Omega_k$ uniformly at random.
14. Else select patient $j \in \Omega_k$ using probability:

$$P_{ij}(d) = \frac{[Q(d,i,j)]^{\alpha}\left[\frac{1}{c_{ij}}\right]^{\beta}}{\sum_{m\in\Omega_k}[Q(d,i,j)]^{\alpha}\left[\frac{1}{c_{ij}}\right]^{\beta}}.$$

15. Update nurse routing state, synchronize multi-nurse teams if required, and track rewards.
16. Record transition tuple (S, j, R, S') into schedule trajectory $\mathcal{T}_k$.
17. End While.
18. End For.
19. Evaluate objective function $F(K_k) = \sum_{p\in P} Y_p$ and append K_k to population V.
20. End For.
21. Sort population V in descending order of objective value and structural feasibility.
22. Partition into elite population V_{elite} ($M_{elite} = \lceil \rho_{elite} M \rceil$) and feasible population $V_{feasible}$ ($M_{feasible} = \lceil \rho_{feasible} M \rceil$).
23. Update historical best solution: If $\max(F(K_k)) > F(K^*)$ then $K^* \leftarrow \arg\max(F(K_k))$.
24. For each population tier $C \in \{V_{elite}, V_{feasible}\}$ do.
25. Assign scaling coefficient ω ($\omega_{elite} > \omega_{feasible}$).
26. For each schedule $K \in C$ and recorded transition $(S, j, R, S') \in \mathcal{T}$ do.
27.

$$Q(d,i,j) \leftarrow Q(d,i,j) + \gamma\omega\left[R + \lambda \max_{k\in\Omega'} Q(d,j,k) - Q(d,i,j)\right].$$

28. End All for loop.
29. Return K^*.

3 Results and Discussion

To evaluate the performance and scalability of the proposed AHMQL, computational experiments were conducted across various patient dimensions. The algorithmic performance is validated by analyzing both fitness function and CPU Time against an exact MILP solver. The comparative computational results are summarized in Table 1.

Table 1. AHMQL and MILP Results.

	MILP		AHMQL		
Dimension	Optimal Obj. Value	CPU Time	Best Obj. Value Found	Training CPU Time (s)	Fulfillment Ratio (%)
30	17	64.95	18	2,502.3	60.00
35	23*	36,000	29	2,845.0	82.86
40	34*	36,000	34	4,052.3	85.00
50	42	124.04	39	4,906.0	78.00
100	89*	36,000	83	13,992.4	83.00

*We set 36,000 sec as the stopping criterion for MILP.

Computational experiments across patient dimensions ($|P| \in \{30, 35, 40, 50, 100\}$) validate the performance of the AHMQL algorithm against an exact MILP solver. As summarized in Table 1, AHMQL scales effectively, maintaining high objective function values with stable constraint-fulfillment ratios between 78% and 85% on medium-to-large instances ($|P| \geq 35$). Conversely, while the exact MILP solver finds optimal schedules for small instances ($|P| = 30$) in 64.95 seconds, it experiences combinatorial explosion in larger dimensions due to strict multi-nurse synchronization and time-window constraints, completely stalling at the 36,000-second execution limit. Crucially, the higher CPU runtimes reported for AHMQL represent the algorithm's offline training phase, where exploration overhead is required to mature the dynamic Q-table. Because routing strategies are completely learned during this phase, real-world deployment on operational testing instances is expected to be nearly instantaneous, requiring only localized exploitation of the pre-trained matrix.

4 Conclusion

This study successfully developed an AHMQL that integrates a rank-based survival selection routine with temporal-difference Q-learning to solve the highly constrained HHCRSP. By replacing static heuristic choices with an adaptive spatial-temporal memory layer, the framework effectively guides solution construction through tightly constrained routing spaces. The experimental results validate the practical viability of this approach, demonstrating high and stable patient fulfillment ratios on large-scale instances where exact mathematical solvers experience combinatorial explosion and fail to converge.

While the validation of overlapping operational constraints causes a nonlinear CPU runtime expansion during the algorithm's offline training phase, this computational investment builds a robust decision matrix. Future work will focus on transitioning this trained framework from the experimental training phase into real-world operational deployment using live datasets, where schedule generation is expected to be executionally rapid during the testing and inference stage.

Acknowledgement This research was partially funded by National Science and Technology Council (Taiwan): NSTC 113-2221-E-155-044-MY3.

References

1. Wu, T.-Y., Majeed, A., Kuo, K.N.: An Overview of The Healthcare System in Taiwan. London J. Prim. Care (Abingdon) **3**(2), 115–119 (2010)
2. Cheng, B., Chang, H., Lin, M., Chen, T., Chou, L., Hwang, S.: Rural-Urban Disparities in Family Physician Practice Patterns: A Nationwide Survey in Taiwan. Int. J. Health Plann. Manage. **34**(1), e464–e473 (2019)
3. Lin, Y.-Y., Huang, C.-S.: Aging in Taiwan: Building a Society for Active Aging and Aging in Place. Gerontologist **56**(2), 176–183 (2016)
4. Lin, M., Ma, L., Ying, C.: Matching Daily Home Health-Care Demands with Supply in Service-Sharing Platforms. Transp. Res. E Logist. Transp. Rev. **135**, 102177 (2021)
5. Bazirha, M., Kadrani, A., Benmansour, R.: Stochastic Home Health Care Routing and Scheduling Problem with Multiple Synchronized Services. Ann. Oper. Res. **320**(2), 573–601 (2023)
6. Zhu, M., Wang, Y., Pu, Z., Hu, J., Wang, X., Ke, R.: Safe, Efficient, and Comfortable Velocity Control Based on Reinforcement Learning for Autonomous Driving. Transp. Res. Part C Emerg. Technol. **117**, 102662 (2020)
7. Qiu, H., Wang, D., Yin, Y., Cheng, T.C.E., Wang, Y.: An Exact Solution Method for Home Health Care Scheduling with Synchronized Services. Naval Research Logistics **69**(5), 715–733 (2022)
8. Liang, Y.-C., Cuevas Juarez, J.R.: A Novel Metaheuristic for Continuous Optimization Problems: Virus Optimization Algorithm. Engineering Optimization **48**(1), 73–93 (2016)

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Branch-Price-and-Cut and Symmetry Reductions for Sports Tournament Scheduling

Jasper van Doornmalen[1][0000−0002−2494−0705] and David Van Bulck[2][0000−0002−1222−4541]

[1] Institute for Mathematical and Computatonal Engineering, Pontificia Universidad Católica de Chile, Chile, mvandoornmalen@uc.cl
[2] Faculty of Economics and Business Administration, Ghent University, Belgium
david.vanbulck@ugent.be

Keywords: Branch-Price-and-Cut · Symmetry Handling · OR in Sports · Sports Timetabling · First-Break-Then-Schedule

1 Introduction

First-Break-Then-Schedule (FBTS, cf. Nemhauser and Trick [7]) is a method for optimizing sports tournament schedules that has gained widespread recognition in the literature for its effectiveness (e.g., [1,5,9]). The method decouples two core decisions: *(i)* for each team, whether they play home (H) or away (A) in each round, i.e., the *home-away pattern* (HAP); and *(ii)* which teams are meeting in which round, i.e., the *opponent schedule*. FBTS faces two critical challenges. First, the method relies on full enumeration of possible HAPs or considers heuristic approaches. On the one hand, full enumeration would give optimal solutions, but for a round set $\mathcal{R}$ and under no further constraints the set of HAPs $\{\mathsf{H},\mathsf{A}\}^{\mathcal{R}}$ is exponential in $|\mathcal{R}|$, so full enumeration is only practical for small instances. On the other hand, heuristic restrictions of this set do not provide optimality guarantees. The second challenge is that the method lacks a systematic backtracking mechanism when HAP assignments turn out infeasible.

We provide an integer programming Branch-Price-and-Cut framework that implements FBTS in an effective and exact way. Rather than enumerating all possible HAPs, our framework starts with the most promising patterns and uses a variable pricing approach to generate missing ones. To organize backtracking, we integrate the Branch-and-Price approach with the traditional Benders' decomposition approach from [10]. Moreover, we apply the symmetry-handling method orbitopal fixing to handle symmetries among teams. Whereas our framework is broadly applicable to many classes of problems, in this extended abstract we focus on one particular problem, for which we present effective preliminary results.

2 Problem definition

We apply our framework to the *Constant-distance Traveling Tournament Problem* (C-TTP(3)), which aims to find a double-round robin (DRR) tournament

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

where the total number of times that a team travels is minimized. More specifically:

- Each team plays exactly one match in every round;
- Each team plays at most three consecutive home games and at most three consecutive away games; and
- Matches (i, j) and (j, i) are not scheduled consecutively.

Urritia and Ribeiro [8] show that the objective considered is equivalent to maximizing the number of *breaks* in the tournament, i.e., where a break occurs if a team plays at home (or away) for two consecutive rounds.

To model the C-TTP(3) as a binary program, we introduce the following notation. Let n be an even number, the number of teams. Let $\mathcal{T} = \{1, \dots, n\}$ be the set of teams, $\mathcal{R} = \{1, \dots, 2n-2\}$ the set of rounds, and $\mathcal{M} = \{(i, j) : i, j \in \mathcal{T}, i \neq j\}$ the matches to be scheduled. Given match (i, j), i is said to be the home team and j the away team. The HAP of a team is denoted by elements in $\{\mathsf{H}, \mathsf{A}\}^{\mathcal{R}}$. A *HAP-set* consists of a HAP for each team, $\{\mathsf{H}, \mathsf{A}\}^{\mathcal{T}\times\mathcal{R}}$. Let $\mathcal{P} \subseteq \{\mathsf{H}, \mathsf{A}\}^{\mathcal{R}}$ be the set of patterns that contain precisely $n-1$ home and away games and at most three equal consecutive entries, and let $c_p := \sum_{r\in\mathcal{R}\setminus\{1\}} \mathbb{1}[p_{r-1} = p_r]$ be the number of breaks for HAP $p \in \mathcal{P}$. Note that $|\mathcal{P}|$ is exponential under these constraints, e.g. any concatenation of $n-1$ blocks (HA) and (AH) is contained in $\mathcal{P}$, yielding $|\mathcal{P}| \geq 2^{n-1}$. An exponentially-sized formulation for C-TTP(3) is:

$$\text{Maximize} \sum_{i\in\mathcal{T}} \sum_{p\in\mathcal{P}} c_p z_{i,p}, \tag{1}$$

$$\text{subject to} \sum_{p\in\mathcal{P}} z_{i,p} = 1 \qquad \text{for all } i \in \mathcal{T}, r \in \mathcal{R}, \tag{2}$$

$$\sum_{p\in\mathcal{P}:p_r=\mathsf{H}} z_{i,p} = h_{i,r} \qquad \text{for all } i \in \mathcal{T}, r \in \mathcal{R}, \tag{3}$$

$$\sum_{j\in\mathcal{T}:i\neq j} x_{(i,j),r} = h_{i,r} \qquad \text{for all } i \in \mathcal{T}, r \in \mathcal{R}, \tag{4}$$

$$\sum_{r\in\mathcal{R}} x_{(i,j),r} = 1 \qquad \text{for all } (i, j) \in \mathcal{M}, \tag{5}$$

$$\sum_{j\in\mathcal{T}\setminus\{i\}} (x_{(i,j),r} + x_{(j,i),r}) = 1 \qquad \text{for all } i \in \mathcal{T}, r \in \mathcal{R}, \tag{6}$$

$$\sum_{m\in\{(i,j),(j,i)\}} (x_{m,r-1} + x_{m,r}) \leq 1 \qquad \text{for all } (i, j) \in \mathcal{M}, r \in \mathcal{R} \setminus \{1\}, \tag{7}$$

$$h_{i,r} \in \{0, 1\} \qquad \text{for all } i \in \mathcal{T}, r \in \mathcal{R}, \tag{8}$$

$$x_{(i,j),r} \in \{0, 1\} \qquad \text{for all } (i, j) \in \mathcal{M}, r \in \mathcal{R}, \tag{9}$$

$$z_{i,p} \in \{0, 1\} \qquad \text{for all } i \in \mathcal{T}, p \in \mathcal{P}. \tag{10}$$

In this formulation, $z_{i,p}$ indicates whether i plays according to the HAP p, $h_{i,r}$ indicates whether i plays home in round r, and $x_{(i,j),r}$ indicates whether match (i, j) is played in round r. The objective function maximizes the overall

Table 1: Computational results for C-TTP(3) with n teams. We compare the compact formulation and Benders' (B) decomposition approach (cf. [10, Section 4.1]) that introduce additional variables for counting the breaks, with the pricing approach (P) and additional symmetry handling (S). We report the best dual bound (D), the best integer primal bound (P), and the runtime in seconds (T). A time limit of 3600 s applies.

n	Compact			B			B + P			B + P + S		
	D	P	T	D	P	T	D	P	T	P	D	T
4	17	17	< 1	17	17	< 1	17	17	< 1	17	17	< 1
6	43	43	34	43	43	23	43	43	< 1	43	43	< 1
8	80	80	74	80	80	37	80	80	< 1	80	80	< 1
10	105	134	3600	107	125	3600	124	124	17	124	124	3
12	147	199	3600	152	187	3600	180	181	3600	181	181	3036
14	198	280	3600	202	260	3600	252	252	417	252	252	14
16	257	352	3600	259	348	3600	327	327	535	327	327	297
18	322	—	3600	326	438	3600	414	420	3600	414	417	3600
20	398	—	3600	397	566	3600	520	520	52	520	520	54

number of breaks, Constraint (2) ensures that a HAP is selected for every team, Constraints (3–6) ensure that the result is a DRR tournament consistent with the HAP selection, and (7) enforces the remaining condition.

3 Contributions

We present a Branch-Price-and-Cut framework to solve a wide class of sports tournament scheduling problems and apply this to C-TTP(3) via a dual decomposition of the program above. Our restricted program consists of the h-variables, we apply variable pricing to the $z_{i,p}$-variables, and we apply separation to ensure existence of a feasible solution for the $x_{(i,j),r}$-variables. Furthermore, since the $h_{i,r}$-variables are persistent in the problem, we can apply symmetry-handling methods to those. More specifically, our contributions are the following:

(i) Regarding the pricing problem, although the number of variables $z_{i,p}$ is exponential in n, the pricing problem runs in polynomial time.

(ii) Regarding the separation problem, we follow the method in [10], where a traditional Benders' decomposition approach is employed to eliminate the x-variables. Briskorn [3] provides reasons to believe that a simple Linear Programming (LP) feasibility check for fixed $z_{i,p}$ variables is already a good heuristic to check feasibility of the generated HAP set, which allows for effective separation in many cases. If the LP relaxation fails to prove infeasibility, we use nogood cuts as a fallback, cf. [10].

(iii) Further, as solution quality for C-TTP(3) is invariant under team permutations, we apply dynamic orbitopal fixing (cf. [2,4]) with symmetry preprocessing on the first round to deal with the team symmetries.

(iv) Last, we provide an implementation via plug-ins for SCIP 10 [6]. In Table 1 we present some indicative preliminary results for C-TTP(3).

Acknowledgements Jasper van Doornmalen thanks the support of ANID FONDECYT Postdoctorado Nr. 3250132. David Van Bulck thanks the support of Research Foundation Flanders (FWO) [23AXE35N].

References

1. Alarcón, F., Durán, G., Guajardo, M., Miranda, J., Muñoz, H., Ramírez, L., Ramírez, M., Sauré, D., Siebert, M., Souyris, S., Weintraub, A., Wolf-Yadlin, R., Zamorano, G.: Operations research transforms the scheduling of Chilean soccer leagues and South American world cup qualifiers. Interfaces **47**, 52–69 (2017). https://doi.org/10.1287/inte.2016.0861
2. Bendotti, P., Fouilhoux, P., Rottner, C.: Orbitopal fixing for the full (sub-)orbitope and application to the unit commitment problem. Mathematical Programming **186**(1–2), 337–372 (Dec 2019). https://doi.org/10.1007/s10107-019-01457-1
3. Briskorn, D.: Feasibility of home–away-pattern sets for round robin tournaments. Operations Research Letters **36**(3), 283–284 (May 2008). https://doi.org/10.1016/j.orl.2007.09.009
4. van Doornmalen, J., Hojny, C.: A unified framework for symmetry handling. Mathematical Programming **212**(1–2), 217–271 (Jun 2024). https://doi.org/10.1007/s10107-024-02102-2
5. Durán, G., Gutiérrez, F., Guajardo, M., Marenco, J., Sauré, D., Zamorano, G.: Scheduling the main professional football league of Argentina. INFORMS Journal on Applied Analytics **51**, 361–372 (2021). https://doi.org/10.1287/inte.2021.1088
6. Hojny, C., Besançon, M., Bestuzheva, K., Borst, S., Chmiela, A., Dionísio, J., Eifler, L., Ghannam, M., Gleixner, A., Göß, A., Hoen, A., van der Hulst, R., Kamp, D., Koch, T., Kofler, K., Lentz, J., Maher, S.J., Mexi, G., Mühmer, E., Pfetsch, M.E., Pokutta, S., Serrano, F., Shinano, Y., Turner, M., Vigerske, S., Walter, M., Weninger, D., Xu, L.: The SCIP Optimization Suite 10.0. Technical report, Optimization Online (November 2025), https://optimization-online.org/2025/11/the-scip-optimization-suite-10-0/
7. Nemhauser, G.L., Trick, M.A.: Scheduling a major college basketball conference. Oper. Res. **46**, 1–8 (1998). https://doi.org/10.1287/opre.46.1.1
8. Urrutia, S., Ribeiro, C.C.: Maximizing breaks and bounding solutions to the mirrored traveling tournament problem. Discrete Appl. Math. **154**, 1932–1938 (2006). https://doi.org/10.1016/j.dam.2006.03.030
9. Van Bulck, D., Goossens, D.: First-break-heuristically-schedule: Constructing highly-constrained sports timetables. Operations Research Letters **51**, 326–331 (2023). https://doi.org/https://doi.org/10.1016/j.orl.2023.03.014
10. Van Bulck, D., Goossens, D.: A traditional Benders' approach to sports timetabling. European Journal of Operational Research **307**(2), 813–826 (Jun 2023). https://doi.org/10.1016/j.ejor.2022.10.044

Multi-Contract Integrated Shift Design and Rostering

Lucas Kletzander[0000−0002−2100−7733]

DBAI, TU Wien, Vienna, Austria
lucas.kletzander@tuwien.ac.at

According to the review by Ernst et al. [3], personnel scheduling is typically done in several stages, including demand modelling, shift scheduling, and line-of-work construction (rostering). In shift scheduling or shift design, the goal is to select shifts to use for covering the demand based on a wide range of possible shift start and end times. However, there are typically additional constraints to fulfil. For example, in Minimum Shift Design [2], the number of different shift start and end times in use across the scheduling horizon should be minimized to reduce complexity of the resulting employee schedules.

On the other hand, decisions in this phase can have significant impact in the next phase, which is called rostering. In this phase, specific rosters are designed for individual employees based on constraints about their total working hours, their allowed shifts (e.g., not all employees might be available for night shifts), their specified working days and many other rules. Further, legal and health-related rules need to be respected [4], e.g., regarding consecutive working days, required rest between shifts, and forbidden sequences of shifts.

In most work in literature, these stages are solved consecutively due to the complexity of the overall problem. Typical applications for problems showing both of these stages are call centres, but also nurse scheduling and production [6]. However, integrations of the two stages are rare [7]. On the other hand, in several problem classes the demand is based on certain tasks that need to be covered, e.g., legs in scheduling for public transport or airlines [1]. In these problems, integrations are more common, e.g., for integrated crew and vehicle scheduling, or for integrated scheduling and rostering.

Limitations of the Sequential Approach. Despite having good or even optimal solutions for shift design, it might be very difficult to find good solutions for the rostering stage or even lead to scenarios where no feasible solution can be found. In particular, reacting to spikes in demand promotes the use of short shifts as these are more flexible, but rostering a large number of short shifts for full-time employees can easily lead to scenarios where these do not have enough weekly hours considering the maximum number of shifts per week (e.g., they should work 5 shifts of 8 hours, but most shifts are around 5 hours).

A partial solution is to integrate extra constraints into the shift design phase. Bounds for average shift lengths [5] work well for a homogenous workforce, e.g., setting these constraints to a narrow window around 8 hours would allow good rosters for full time employees in the previous example. However, this technique

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

can fail in more complex examples. One issue might be very uneven demand across days. In this case, constraints on the sequences of consecutive days might be hard to satisfy, e.g. if most of the workforce is required on Monday, but then there is a sharp drop to Tuesday, while at the same time individual working days are not allowed.

Another critical case is the existence of a heterogenous workforce, e.g., a combination of full-time and part-time employees. Restricting the average does not work in this case, as, e.g., a workforce consisting of 50 % full-time and 50 % part-time employees requires an average shift length of around 6 hours but preferably achieved by 8-hour shifts for full-time and 4-hour shifts for part-time employees. On the other hand, a different company might have part-time employees with different contracts, requiring fewer shifts with 6 hours, or even mix multiple types of part-time contracts.

Integrated Problem Formulation. Therefore, we propose an integrated formulation for combined shift design and rostering in the presence of multiple contracts, named the Multi-Contract Integrated Shift Design and Rostering Problem, which covers a real-life use case from a collaboration with the company XIMES GmbH. The problem demand is specified by a time series, giving the required number of employees at each point in time. This could be extended to multiple time series, e.g., also including required skills of employees. Constraints on shifts are specified by a set of shift types, each defining aspects including windows for shift start time, end time, and length, and allowed days. Contracts can specify rules for employees including, but not limited to constraints about:

- Required working time
- Maximum deviation from working time
- Allowed shift types per day
- Minimum and maximum number of shifts per week
- Minimum and maximum consecutive assignments
- Minimum and maximum consecutive rest days
- Forbidden shift sequences
- Minimum rest time between shifts

For each contract, the number of employees available under the contract is specified. For optimization, the goal is to balance the objectives taken from shift design (minimize overcover, minimize undercover, minimize number of different shift start and end times) with optional soft constraints on the contracts (e.g., there might be a hard minimum number of consecutive assignments of 2, a soft minimum of 3, or both of them at once).

Solution Approach. Based on the combination of rules from two different problems, the search space can be very large and include a large number of competing objectives to optimize, on the other hand, there is the potential for substantial gains in solution quality over the sequential approach. The current solution approach uses Simulated Annealing with multiple neighbourhoods:

- Add / remove / replace shift
- Move shift within / across contract
- Move multiple shifts within contract
- Replace all shifts with certain start and end time

Preliminary results show that the approach is able to provide high-quality solutions to real-life and real-life-based instances scheduling up to several hundred employees for a week that closely cover the demand while satisfying the contract-based constraints. Results also show a clear benefit over the sequential approach using the same solution method in two stages in scenarios with different contracts, validating the decision to solve the problem in an integrated style. Future work will include a more detailed evaluation of different moves, developing other solution approaches, and evaluating the approaches in more varied scenarios with different contract and demand structures.

References

1. Abbas, A.K., Yassen, E.T.: Crew scheduling problems: A systematic literature review. IEEE Access **13**, 190196–190220 (2025)
2. Di Gaspero, L., Gärtner, J., Kortsarz, G., Musliu, N., Schaerf, A., Slany, W.: The minimum shift design problem. Annals of operations research **155**(1), 79–105 (2007)
3. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Sier, D.: Staff scheduling and rostering: A review of applications, methods and models. European journal of operational research **153**(1), 3–27 (2004)
4. Gärtner, J., Bohle, P., Arlinghaus, A., Schafhauser, W., Krennwallner, T., Widl, M.: Scheduling matters-some potential requirements for future rostering competitions from a practitioner's view. In: PATAT 2018-Proceedings of the 12th international conference on the practice and theory of automated timetabling. pp. 33–42 (2018)
5. Kletzander, L., Musliu, N.: Hyper-heuristics for personnel scheduling domains. Artificial Intelligence **334**, 104172 (2024)
6. Özder, E.H., Özcan, E., Eren, T.: A systematic literature review for personnel scheduling problems. International Journal of Information Technology & Decision Making **19**(06), 1695–1735 (2020)
7. Türker, T., Demiriz, A.: An integrated approach for shift scheduling and rostering problems with break times for inbound call centers. Mathematical Problems in Engineering **2018**(1), 7870849 (2018)

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Large Language Model-Based Explainable Hyper-heuristics for Examination Timetabling

Nilgün Şengöz[1,2⋆], Ender Özcan[2], and Jeremie Clos[2]

[1] Burdur Mehmet Akif Ersoy University, Burdur, Türkiye
nilgunsengoz@mehmetakif.edu.tr
[2] University of Nottingham, Nottingham, UK
{ender.ozcan,jeremie.clos}@nottingham.ac.uk

Abstract. We introduce LLM-STAR (Structure-Aware Selector), a generation hyper-heuristic that uses GPT-4 as an offline knowledge source to produce a transparent, deterministic rule engine for algorithm selection in examination timetabling. The approach is motivated by the observation that no single graph colouring heuristic dominates across the structural diversity of realistic timetabling conflict graphs, and that human-readable selection rules are preferable to opaque learned models in scheduling practice. Using GPT-4 in an offline distillation phase, processing selected academic papers on examination timetabling and graph colouring, we convert chain-of-thought reasoning into a seven-rule engine that maps five structural instance features to candidate heuristic sets. Experiments on the Toronto benchmark instances demonstrate a perfect success rate (13/13). Independent validation on 18 randomly generated instances confirms out-of-sample generalisation (18/18), demonstrating the viability of offline LLM distillation for explainable combinatorial optimisation.

Keywords: Timetabling · Graph colouring · Hyper-heuristics · Large language models · Algorithm selection

1 Introduction

The algorithm selection problem [9] remains a central challenge in combinatorial optimisation, requiring prediction of the best-performing algorithm for a given instance based on measurable features. Examination timetabling is a widely studied NP-hard problem [8], reducible to the Minimum Graph Colouring Problem [6] in its simplest form, where each vertex represents an examination and edges indicate conflicts arising from shared student enrolments. The objective of assigning the minimum number of time-slots to a set of mutually non-clashing exams is therefore equivalent to determining the chromatic number of the corresponding graph. Well-known constructive heuristics such as Welsh-Powell, DSATUR, and

⋆ Corresponding author.

Smallest-Last build feasible but near-optimal solutions efficiently [2]; however, no single heuristic consistently dominates across diverse graph structures.

A key aspect of LLM-STAR is that it targets the *structural subspace* of graph colouring that corresponds to realistic examination timetabling instances, rather than the full space of all graphs. Examination conflict graphs are not arbitrary: student enrolment patterns are shaped by curriculum structure and departmental boundaries, producing local clustering that is absent in random graphs of comparable density. This yields conflict graphs with elevated clustering coefficients (κ) and degree variance profiles (σ^2_δ) reflecting programme-level modular structure, while edge density ρ is bounded by realistic enrolment distributions. The five structural features $(\rho, \kappa, \sigma^2_\delta, \delta_{\max}, n)$ capture the axes along which realistic timetabling instances vary [3] and along which heuristic performance differences are most pronounced. This work addresses algorithm selection within this timetabling-relevant subspace and does not claim generality across all graph colouring instances.

Recent studies explore large language models (LLMs) as online optimisation engines [12] and evolutionary code generators [14]. In contrast, this study investigates LLMs as an *offline* knowledge source for generation hyper-heuristics [5]. We introduce LLM-STAR, which uses GPT-4 to distil academic literature into a stand-alone, explainable rule engine; once generated, the engine operates entirely without the LLM as a deterministic, auditable decision procedure.

Relation to Instance Space Analysis. Instance Space Analysis (ISA) [11] provides a systematic methodology for understanding algorithm performance across a feature-defined instance space, and has been applied to graph colouring [10] and curriculum-based course timetabling [4]. LLM-STAR shares the goal of mapping instance structure to algorithm performance but differs from ISA in three respects: (i) features are derived from LLM-distilled domain expertise in the literature rather than by automated statistical selection; (ii) decision boundaries are produced by chain-of-thought reasoning and maximum-margin thresholding rather than geometric footprint analysis; and (iii) the output is a set of natural-language-annotated deterministic rules deployable without visualisation infrastructure, rather than a footprint map. The two approaches are complementary: ISA methodology could validate LLM-STAR's feature space and thresholds, which we identify as future work.

InstSpecHH [15] generates instance-specific heuristics via LLM meta-feature interpretation. HeurAgenix [13] introduces a two-phase framework with offline heuristic evolution followed by online LLM-guided selection; unlike LLM-STAR, it still invokes the LLM at runtime, incurring latency and non-determinism. Atias [1] applied LLMs to multi-objective timetabling, confirming their promise in scheduling domains.

The remainder of this paper is structured as follows: Section 2 details the LLM-STAR architecture and the LLM knowledge distillation protocol. Section 3 presents the experimental results, statistical analysis, and independent validation. Section 4 concludes the study.

2 Rule Engine Generation for Heuristic Selection

GPT-4 (gpt-4-0613 API) is initially provided with selected academic papers on examination timetabling and graph colouring to identify important features for algorithm selection. This offline training phase is followed by a four-stage process building a rule-based engine.

Stage 1. Instances are characterised using a five-dimensional structural feature vector $\mathbf{f} = (\rho, \kappa, \sigma_\delta^2, \delta_{\max}, n)$:

- **Edge density** ρ: the proportion of realised edges relative to the maximum possible:
$$\rho = \frac{|E|}{\binom{n}{2}} = \frac{2|E|}{n(n-1)} \tag{1}$$
- **Global clustering coefficient** κ (transitivity [7]): the ratio of closed triplets (triangles) to *all* connected triplets, measuring the tendency of conflict neighbourhoods to form cliques:
$$\kappa = \frac{3 \times |\triangle(G)|}{\sum_{v \in V} \binom{\delta_v}{2}} \tag{2}$$
where $|\triangle(G)|$ is the number of triangles, δ_v is the degree of vertex v, and $\binom{\delta_v}{2}$ counts the connected triples centred at v. This is the *global* (transitivity) form of [7], distinct from the average local clustering coefficient.
- **Degree variance** σ_δ^2 (population variance):
$$\sigma_\delta^2 = \frac{1}{n} \sum_{v \in V} (\delta_v - \bar{\delta})^2, \quad \bar{\delta} = \frac{2|E|}{n} \tag{3}$$
- **Maximum degree** $\delta_{\max} = \max_{v \in V} \delta_v$.
- **Instance size** $n = |V|$: the number of examinations.

Figure 1 illustrates these features on a small example conflict graph.

These features capture computational difficulty [3] in graph colouring and timetabling. Features are extracted from the Toronto benchmark instances [8]; these instances are used throughout all stages of our workflow and are treated as the effective training set.

Stage 2. GPT-4 is prompted five times per instance, using structural features and observed algorithm performance from academic studies on the Toronto training instances, to produce chain-of-thought responses identifying promising heuristic subsets from a pool of five algorithms: {WP, DSA, SL, BFS, DFS}. Prompts follow the structure: *"Given feature vector* $\mathbf{f}$ *and performance data showing [heuristic rankings], which 1–2 heuristics are most likely to succeed? Explain your structural reasoning."* All 65 raw chain-of-thought responses will be made publicly available, enabling independent verification of per-instance

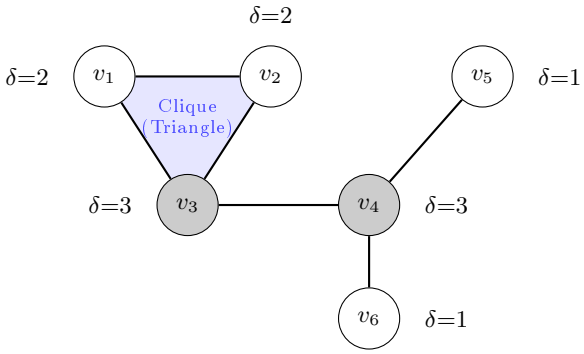


Fig. 1. Five structural features illustrated on a toy conflict graph with n=6 examinations and $|E|$=6 conflict edges. The shaded triangle (v_1, v_2, v_3) is the graph's only triangle and the sole contributor to the clustering coefficient, while the grey vertices v_3 and v_4 attain the maximum degree. For this graph the features evaluate to an edge density ρ=$2|E|/[n(n-1)]$=0.40 (Eq. 1), a global clustering coefficient κ=3/8=0.375 (Eq. 2, transitivity [7]), a degree variance σ_δ^2=0.67 about a mean degree $\bar{\delta}$=2.0 (Eq. 3), and a maximum degree $\delta_{\max}$=3. We emphasise that κ denotes the *global* (transitivity) coefficient of [7]; the average local clustering coefficient yields a distinct value (0.39) that is not employed by the rule engine.

consensus levels and LLM consistency across repeated queries. All source papers processed in this offline phase were accessed lawfully through institutional subscriptions or open-access venues and were used solely for non-consumptive text-and-data-mining for scientific research; no verbatim text from these sources is reproduced in the rule engine or in this paper.

Stage 3. A recommendation is accepted if at least four of five responses agree. The LLM provided qualitative rationales (e.g., *"sparse and highly clustered, favouring degeneracy ordering"*) based on continuous feature values. To convert instance-specific rationales into generalisable rules, we project training instances onto the five-dimensional feature space, identifying three structural classes through manual inspection: heavy-tailed degree distributions, sparse clustered, and medium-density unstructured.

Stage 4. We formalise quantitative decision boundaries through comparative threshold analysis between instances where the LLM preferred different heuristic classes. The quantitative boundaries (τ) dividing the five-dimensional feature space $\mathcal{F}$ into the seven operational rules (Table 1) are defined as follows.

Density boundaries (τ_ρ). The transition from sparse to medium-density to dense graphs is partitioned at τ_{ρ_1}=0.12 and τ_{ρ_2}=0.35, calculated via the maximum-margin separation:

$$\tau_\rho = \frac{\max_{x \in C_{\text{sparse}}} \rho(x) + \min_{x \in C_{\text{dense}}} \rho(x)}{2} \tag{4}$$

where C_{sparse} and C_{dense} are instance sets identified by Stage 3 consensus.

Table 1. LLM-generated Rule Engine: seven structure-conditioned selection rules (priority order). DSA=DSATUR; SL=Smallest-Last; WP=Welsh-Powell; BFS=BFS-Sequential.

#	Structural Condition	Candidate Set	LLM-Distilled Rationale
1	$n<300$ & $\rho>0.35$	{DSA, SL}	Dense small graph; saturation and degeneracy competitive.
2	$n\geq400$ & $\sigma_\delta^2>80$ & $\delta_{\max}>3\bar{\delta}$	{DSA, WP}	Heavy-tailed topology; degree-based methods effective.
3	$\rho\leq0.12$ & $\kappa>0.40$	{SL, DSA}	Sparse clustered; degeneracy exploits dense subgraphs.
4	$0.12<\rho\leq0.35$ & $\kappa>0.30$	{DSA, SL}	Medium-density clustered; saturation dominant.
5	$n\geq300$ & $\rho\leq0.10$ & $\sigma_\delta^2<20$	{SL, BFS}	Large sparse uniform; traversal beats saturation.
6	$100\leq n<400$ & $0.10<\rho\leq0.20$	{BFS, DSA}	Medium-scale unstructured; BFS captures neighbourhood structure.
7	(default)	{DSA}	Default fallback for ambiguous signatures.

Clustering boundaries (τ_κ). To distinguish random edge distributions from clustered enrolment patterns, the clustering margin is empirically set at $\tau_\kappa \in \{0.30, 0.40\}$ using the same maximum-margin equation.

Heavy-tailed detection. An instance triggers Rule 2 if:

$$\delta_{\max} > 3\bar{\delta} \quad \wedge \quad \sigma_\delta^2 > 80 \tag{5}$$

The threshold $\sigma_\delta^2>80$, together with the $\delta_{\max}>3\bar{\delta}$ condition, isolates the single extreme heavy-tailed instance (pur-s-93, σ_δ^2=8527.5); other high-variance instances (e.g. car-s-91, σ_δ^2=3843.7) do not satisfy the joint condition. We explicitly acknowledge that this single-instance calibration presents an overfitting risk; validation on additional heavy-tailed topologies is essential to confirm generalisation.

Scale boundaries (τ_n). Instance size thresholds ($\tau_n \in \{300, 400\}$) delineate scale regimes where the asymptotic behaviour of specific heuristics becomes relevant.

This formalisation ensures that the transition from five continuous features to seven discrete rules is both structurally grounded in LLM reasoning and quantitatively calibrated to the benchmark distribution, replacing ad-hoc parameter tuning with benchmark-calibrated thresholds.

The rule engine operates in two steps: (i) evaluate the feature vector against rules in priority order; the first match returns the candidate set. (ii) execute only the candidate heuristics and select the minimum-timeslot solution.

3 Experimental Results

Table 2 summarises LLM-STAR performance on 13 Toronto benchmark instances [8] (training set) and 18 randomly generated instances (validation set). The validation set was produced using Qu et al.'s generator [8], with a stratified design of 9 small (n<100) and 9 large (n≥500) instances, with densities spanning 6%–48% to broadly mirror the Toronto distribution. This split ensures that rules gated on scale boundaries (Rules 1, 2, 5, 6) are tested in both size regimes. We acknowledge that the density range does not cover extreme sparsity (ρ<5%) or very high density (ρ>60%); these represent directions for future stress-testing.

LLM-STAR achieves a success rate—defined as the proportion of instances matching empirical pool-best performance (best among the five constructive heuristics)—of 100% on Toronto (13/13). Statistical analysis using the Wilcoxon signed-rank test (two-tailed) confirms that LLM-STAR selections significantly outperform a static Smallest-Last baseline: W=0.00, p=0.002, Cohen's d=−1.05 [95% CI: $-1.70, -0.71$], indicating a large effect size. Smallest-Last serves as baseline following prior comparative work [2]; note that Rule 7 designates DSATUR as the default fallback, and comparison against static DSATUR is reserved for the extended version where per-heuristic timeslot data will be reported in full.

To address overfitting concerns, LLM-STAR was validated on 18 randomly generated out-of-sample instances [8]: 9 small (n<100), 9 large (n≥500), densities spanning 6%–48%. LLM-STAR achieved full success across all instances, matching pool-best on every validation instance. Four rules activated {R1, R4, R6, R7}, demonstrating generalisation across unseen instances.

Rule 3 (sparse clustered: $\rho \leq 0.12 \wedge \kappa > 0.40$) successfully identifies structural anomalies like `lse-f-91`, where Smallest-Last outperforms DSATUR (18 vs. 19 timeslots). Rule 4 (medium-density clustered) is the most frequently triggered rule (7/13 Toronto, 6/18 generated), reflecting the prevalence of this structural class. Rule 5 (large sparse uniform) never activates on either benchmark, indicating that neither Toronto nor generated instances sufficiently represent this structural regime; this defines a validation target for future work. The perfect success rate across both training and validation sets confirms that LLM-distilled structural conditions generalise beyond the training distribution—consistent with the generalisation patterns observed in ISA studies on adjacent domains [4,10].

4 Conclusion

This study demonstrates that the reasoning capabilities of large language models can be systematically distilled into deterministic and interpretable rules for guiding heuristic selection. Using examination timetabling as a case study, we showed that the proposed LLM-STAR framework reliably identifies the best-performing graph colouring heuristic within the structural subspace corresponding to realistic timetabling conflict graphs, achieving full success on both the Toronto benchmark and randomly generated validation instances, while providing transparent and auditable explanations for each decision.

Table 2. Performance summary of LLM-STAR. **R** = rule triggered; **PB** = pool-best timeslots (best among five heuristics). **Bold** entries: LLM-STAR matches Pool-Best. Conflict densities ρ are computed via Eq. (1) (Carter et al.); rules are evaluated in priority order (Table 1).

Toronto Benchmark (13/13)				**Generated Instances (18/18)**			
Inst.	ρ	**R**	**PB**	**Inst.**	ρ	**R**	**PB**
car-f-92	0.138	R4	**30**	S-p6	0.062	R7	**8**
car-s-91	0.128	R4	**35**	S-p12	0.125	R4	**10**
ear-f-83	0.267	R4	**24**	S-p18	0.184	R6	**12**
hec-s-92	0.421	R1	**19**	S-p25	0.267	R4	**12**
kfu-s-93	0.056	R3	**19**	S-p32	0.335	R4	**15**
lse-f-91	0.063	R3	**18**	S-p38	0.398	R1	**16**
pur-s-93	0.029	R2	**35**	S-p42	0.435	R1	**17**
rye-s-93	0.075	R3	**23**	S-p45	0.475	R1	**17**
sta-f-83	0.144	R4	**13**	S-p48	0.502	R1	**18**
tre-s-92	0.181	R4	**23**	L-p5	0.052	R7	**12**
uta-s-92	0.126	R4	**35**	L-p10	0.102	R6	**22**
ute-s-92	0.085	R3	**10**	L-p15	0.155	R4	**28**
yor-f-83	0.289	R4	**21**	L-p20	0.208	R6	**32**
				L-p25	0.264	R4	**35**
				L-p30	0.315	R4	**38**
				L-p35	0.362	R1	**42**
				L-p40	0.418	R1	**45**
				L-p45	0.468	R1	**48**

It is essential to assess the broader generalisability of the proposed generation hyper-heuristic approach by training and testing on structurally dissimilar benchmark sets (e.g., ITC 2007, Carter benchmarks). As future work, we also plan to expand the candidate set to include metaheuristics and local search operators, to investigate combined ISA–LLM distillation pipelines [11], and to apply the methodology to wider timetabling and scheduling domains without abstracting away their real-world complexities.

References

1. Atias, V., Doychev, E.: LLM implementation for multi-objective university timetabling: technical description. In: Proceedings of ICAI 2025 (2025)
2. Burke, E.K., Qu, R., Soghier, A.: Adaptive selection of heuristics for improving exam timetables. Ann. Oper. Res. **218**(1), 129–145 (2012)

3. Cheeseman, P., Kanefsky, B., Taylor, W.M.: Where the really hard problems are. In: IJCAI-91, pp. 331–337 (1991)
4. De Coster, A., Musliu, N., Schaerf, A., Schoisswohl, J., Smith-Miles, K.: Algorithm selection and instance space analysis for curriculum-based course timetabling. J. Scheduling **25**(1), 35–58 (2022)
5. Drake, J.H., Kheiri, A., Özcan, E., Burke, E.K.: Recent advances in selection hyper-heuristics. Eur. J. Oper. Res. **285**(2), 405–428 (2020)
6. Lewis, R.M.R.: A Guide to Graph Colouring: Algorithms and Applications. Springer (2016)
7. Newman, M.E.J.: The structure and function of complex networks. SIAM Rev. **45**(2), 167–256 (2003)
8. Qu, R., Burke, E.K., McCollum, B., Merlot, L.T.G., Lee, S.Y.: A survey of search methodologies and automated system development for examination timetabling. J. Scheduling **12**(1), 55–89 (2009)
9. Rice, J.R.: The algorithm selection problem. Adv. Comput. **15**, 65–118 (1976)
10. Smith-Miles, K., Baatar, D., Wreford, B., Lewis, R.: Towards objective measures of algorithm performance across instance space. Comput. Oper. Res. **45**, 12–24 (2014)
11. Smith-Miles, K., Muñoz, M.A.: Instance space analysis for algorithm testing: methodology and software tools. ACM Comput. Surv. **55**(12), 1–38 (2023)
12. Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q.V., Zhou, D.: Large language models as optimisers. arXiv:2309.03409 (2023)
13. Yang, X., Zhang, L., Qian, H., Song, L., Bian, J.: HeurAgenix: Leveraging LLMs for solving complex combinatorial optimization challenges. arXiv:2506.15196 (2025)
14. Ye, H., Wang, J., Cao, Z., Berto, F., Hua, C., Kim, H., Park, J., Song, G.: ReEvo: Large language models as hyper-heuristics with reflective evolution. In: Advances in Neural Information Processing Systems (NeurIPS) (2024)
15. Zhang, S., Liu, S., Lu, N., Wu, J., Liu, J., Ong, Y.S., Tang, K.: LLM-driven instance-specific heuristic generation and selection. IEEE Trans. Emerg. Top. Comput. Intell. (2026). arXiv:2506.00490

Large Neighborhood Search for Generalized Parallel Machine Scheduling

Lukas Frühwirth, Nysret Musliu, and Felix Winter

DBAI, TU Wien, Vienna, Austria
{lukas.fruehwirth, nysret.musliu, felix.winter}@tuwien.ac.at

Keywords: Large Neighborhood Search · Parallel Machine Scheduling · Constraint Programming · Rich Scheduling · Production Scheduling

1 Introduction

Real-world production scheduling problems often combine constraints from multiple classical formulations such as Unrelated Parallel Machine Scheduling, Flexible Job Shop Scheduling, and Resource-Constrained Project Scheduling (RCPSP). We recently introduced the Generalized Parallel Machine Scheduling (GPMS) framework [1], which assigns jobs to unit-capacity or non-capacitated machines, minimizing a weighted sum of objectives such as tardiness, makespan, and setup time. The framework unifies machine eligibility, sequence- and machine-dependent setup times, machine calendars, precedence relations with time lags, and secondary resources with time-varying capacities and pulse and step demands. A complete formal definition is available in [1].

While the proposed constraint programming (CP) models solve small-to-medium GPMS instances effectively, scalability remains challenging: instances with more than 20 jobs frequently cannot be solved to optimality within practical time limits, and complex constraints such as secondary resources with step demands increase model size significantly, limiting improvements over heuristic solutions even on smaller instances.

This scaling limitation motivates the development of a metaheuristic approach. Large Neighborhood Search (LNS) iteratively destroys and repairs parts of a solution, exploring large neighborhoods that would be intractable for classical local search [7, 6]. LNS with CP-based repair has proven particularly effective in the scheduling domain, achieving state-of-the-art results for single-mode scheduling [3], RCPSP variants [8], and flexible job shop scheduling [2, 4]. However, the specific operator design for the full GPMS constraint set remains an open challenge.

In this work, we propose an LNS framework for GPMS with domain-specific neighborhoods, CP-based and heuristic repair operators, and various search strategies. We present preliminary experimental results indicating the effectiveness of the approach.

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Algorithm 1 LNS for GPMS

```
σ ← σ_0
while elapsed time < t_max do
    D ← select a subset of jobs to destroy (neighborhood 𝒩, strategy 𝒮)
    Construct sub-instance ℐ′ from D and σ
    σ′_D ← ℛ(ℐ′, t_repair)
    if σ′_D feasible ∧ obj(Merge(σ, σ′_D)) ≤ obj(σ) then
        σ ← Merge(σ, σ′_D);
return σ
```

2 Large Neighborhood Search Framework

Our LNS approach iteratively improves an initial feasible schedule produced by the randomized construction heuristic (CH) from [1]. Algorithm 1 outlines the core loop: σ_0 is the initial heuristic schedule, σ the current best schedule, and $\mathcal{R}$ the repair operator with time budget t_{repair}. In each iteration, a neighborhood operator selects a subset of jobs D to destroy. A repair operator re-optimizes their assignment and timing, and the result is merged back if it does not worsen the objective (greedy acceptance). The sub-problem can be formulated either by extracting a self-contained sub-instance or by fixing variables in the full CP model. Each run uses a fixed configuration of neighborhood, repair operator, and search strategy; adaptive operator selection is planned for future work.

Neighborhoods. All neighborhoods share a parameter $\bar{n}$ controlling the maximum number of destroyed jobs. The *machine* neighborhood iteratively adds machines (in an order determined by the search strategy) and all their assigned jobs to D until $|D| \geq \bar{n}$, always including at least one machine. The *time-window* neighborhood selects a reference time t based on the search strategy and alternately adds the closest jobs to the left and right of t (by setup start time) until $|D| = \bar{n}$. The *sequence* neighborhood destroys a contiguous block of up to $\bar{n}$ jobs on a single machine, targeting fine-grained resequencing. The *random* neighborhood uniformly samples $\bar{n}$ jobs, serving as a diversification baseline.

Sub-Instance Construction. The destroyed jobs and their affected machines are extracted into a self-contained GPMS sub-instance. Boundary conditions, such as sequence-dependent setups, machine calendars, secondary-resource capacities, and precedence relations, are encoded into the sub-instance so that any feasible sub-solution can be merged back into the global schedule.

As an alternative, a *variable-fixing* strategy fixes all jobs outside D in the full CP model rather than extracting a sub-instance. This avoids the complexity of boundary handling but incurs higher presolving overhead.

Repair Operators. Each repair operator works within a time budget t_{repair}. The *CP-SAT* repair uses the interval-variable model from [1], solved with Google's CP-SAT solver [5], warm-started with the current sub-schedule. It can operate

on either an extracted sub-instance or on the full model with variable fixing. The *construction heuristic* repair applies the randomized heuristic from [1] with top-k selection for diversification and only operates on the extracted sub-instance.

Search Strategies. A *random* strategy selects neighborhood parameters uniformly at random. A *sliding* strategy systematically traverses the schedule, cycling through machines or partitioning jobs into temporal groups of size $\bar{n}$. An *objective-guided* strategy prioritizes schedule regions with high penalty contributions, e.g., selecting machines with the highest tardiness or time windows containing the costliest setups.

3 Preliminary Results

Experimental Setup. All experiments were run with $t_{\max} = 600\,\text{s}$ and $t_{\text{repair}} = 30\,\text{s}$. The maximum destroy size was set to $\bar{n} = 40$ for instances 1–5 and $\bar{n} = 10$ for instances 6–10.

Instances. We evaluated on 10 GPMS benchmark instances from [1]. Instances 1–5 are real-world instances: 1–3 are pure unrelated parallel machine problems with sequence-dependent setups, and 4–5 include precedence relations and secondary resources. Instances 6–10 are generated instances with 20 to 50 jobs, featuring complex precedence graphs with minimum/maximum time lags and conditional eligibilities, as well as secondary resources with pulse and step demands.

Evaluated Configurations. Table 1 reports results for eight configurations. Since CH from [1] is currently the best-performing heuristic for large and heavily constrained GPMS instances, we report all objective values relative to the CH solution. IV_w represents the interval-variable CP-SAT model warm-started with CH. The seven LNS variants are labeled N-P-S, where N denotes the neighborhood (RD = random, M = machine, T = time-window), P the sub-problem strategy (VF = variable fixing, SI = sub-instance extraction), and S the search strategy (R = random, S = sliding). All LNS variants use the warm-started IV_w model as repair operator.

Discussion. These results are preliminary, and we draw no firm conclusions from them at this stage. The overall tendency is that LNS improves over the standalone IV_w model, with the gap widening on larger instances. On instances 3–5 ($|J| \geq 1167$), IV_w and all variable-fixing variants fail to improve over CH, while sub-instance-based configurations reduce the objective to as low as 0.285 of the baseline. On the small generated instances (6–7), IV_w already performs well, finding the optimum on instance 6, and several LNS variants match its results. On the larger generated instances (8–10), IV_w fails to improve over CH, whereas LNS variants achieve improvements of up to 15%. No single configuration dominates, however: the best results are spread across different neighborhoods and search strategies. This indicates that LNS is a promising direction for GPMS,

Table 1. Objective values relative to the CH baseline ($t_{\max} = 600\,\text{s}$). Values below 1 indicate improvement over CH; lower is better. Best per instance in bold.

Inst.	$\lvert J\rvert$	IV_w	RD-VF-R	M-VF-R	M-SI-R	M-SI-S	T-VF-R	T-SI-R	T-SI-S
1	72	0.540	0.567	0.554	—	—	**0.538**	0.543	0.561
2	601	0.977	0.969	0.899	—	—	0.755	0.619	**0.499**
3	1167	1.000	1.000	1.000	0.408	**0.285**	1.000	0.764	0.697
4	1694	1.000	1.000	1.000	1.000	1.000	1.000	**0.959**	0.962
5	2142	1.000	1.000	1.000	1.000	1.000	1.000	0.989	**0.959**
6	20	**0.864**	**0.864**	**0.864**	**0.864**	**0.864**	0.910	0.910	0.922
7	20	0.835	0.835	**0.834**	**0.834**	0.838	0.835	0.838	0.957
8	50	1.000	1.000	1.000	0.979	0.978	1.000	**0.960**	0.980
9	50	1.000	0.987	1.000	0.996	0.996	**0.980**	0.981	0.984
10	50	1.000	0.980	1.000	**0.852**	**0.852**	0.989	1.000	0.978

but that no fixed variant can yet be regarded as generally preferable, which motivates the adaptive operator selection we leave for future work. With only ten instances, these observations remain indicative rather than conclusive, and a reliable assessment will require the substantially larger and more diverse instance set we plan to evaluate in the extended version of this study.

References

1. Frühwirth, L., Einspieler, C., Musliu, N., Winter, F.: GPMS: A generalized parallel machine scheduling framework with rich temporal and resource constraints. In: Proceedings of the 36th International Conference on Automated Planning and Scheduling (ICAPS) (2026), preprint: https://tucloud.tuwien.ac.at/index.php/s/8ZqHFkt75bPFLDc
2. Kasapidis, G.A., Paraskevopoulos, D.C., Mourtos, I., Repoussis, P.P.: A unified solution framework for flexible job shop scheduling problems with multiple resource constraints. European Journal of Operational Research **320**(3), 479–495 (2025). https://doi.org/10.1016/j.ejor.2024.08.010
3. Laborie, P., Godard, D.: Self-adapting large neighborhood search: Application to single-mode scheduling problems. In: Proceedings of the 3rd Multidisciplinary International Scheduling Conference (MISTA). pp. 276–284 (2007)
4. Pacino, D., Van Hentenryck, P.: Large neighborhood search and adaptive randomized decompositions for flexible jobshop scheduling. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Three. p. 1997–2002. IJCAI'11, AAAI Press (2011)
5. Perron, L., Didier, F., Gay, S.: The CP-SAT-LP solver. In: 29th International Conference on Principles and Practice of Constraint Programming (CP 2023). Leibniz International Proceedings in Informatics (LIPIcs), vol. 280, pp. 3:1–3:2. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2023). https://doi.org/10.4230/LIPIcs.CP.2023.3
6. Pisinger, D., Ropke, S.: Large neighborhood search. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics, pp. 99–127. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-319-91086-4˙4

7. Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. In: Principles and Practice of Constraint Programming (CP 1998). LNCS, vol. 1520, pp. 417–431. Springer (1998). https://doi.org/10.1007/3-540-49481-2˙30
8. Vilím, P., Laborie, P., Shaw, P.: Failure-directed search for constraint-based scheduling. In: Michel, L. (ed.) Integration of AI and OR Techniques in Constraint Programming. pp. 437–453. Springer International Publishing, Cham (2015)

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Untis Course Optimization for Finnish High Schools

Lukas Matthias Wolf

Untis GmbH, Belvederegasse 11, 2000 Stockerau, Austria
lukas.m.wolf@untis.at
http://www.untis.at

Abstract. Untis has provided scheduling software for schools worldwide for over 50 years. In recent years, the optimization team at Untis, in collaboration with IMC Krems University of Applied Sciences, has developed new approaches to course optimization. In this talk, we present one such approach: a course optimization for Finnish high schools. The solution is currently deployed and in operational use across multiple Finnish schools.

Finnish high schools operate under a distinctive timetabling regime: a fixed master timetable, shared across all schools in a district, assigns time slots to clusters across multiple terms. A cluster is a set of courses that take place at the same time. Because time slots are aligned district-wide, students can attend courses at neighboring schools during the first and last periods of the day — a feature rarely seen in other countries. These circumstances significantly shape the resulting optimization problem.

The course optimization problem in this setting is a variant of student sectioning: given a set of students with their course choices, a set of courses, terms, and a fixed number of clusters, assign each student to set of courses and each course to clusters and terms so as to minimize the total number of student conflicts, subject to constraints such as:

- Each student is assigned to at most one cluster per time slot (no conflicts).
- Each teacher is assigned to at most one cluster per time slot. Teachers are preassigned to courses.
- Each room type has a limited number of rooms available per time slot, and each course requires specific room types.
- Each course must be assigned to exactly one cluster per term it takes place in.
- A course spanning multiple terms must be assigned to clusters in consecutive terms.
- Certain courses must be completed by a student in a prescribed order (prerequisite constraints).
- Certain courses must be assigned to the same clusters.

We formulate the problem as a mixed-integer program (MIP). However, real-world instances with up to 1500 students and 5-6 terms per academic year cannot be solved directly by state-of-the-art commercial and open-source MIP solvers (Gurobi, CPLEX, OR-Tools). The main difficulty is the lack of decomposable structure: in practice, student course choices always overlap across class levels and topic groups, preventing

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

partitioning into independent subproblems. Furthermore, dependencies between terms restrict the decomposability.
Our approach combines column generation with large neighborhood search (LNS). We select the initial set of representative students to cover the most constrained cases first, while ensuring that all course types and prerequisite course chain variants are represented. We solve the restricted MIP, analyze the solution to identify violated constraints, and iteratively add students. The LNS phase then improves the solution, considering all students simultaneously. For many real-world instances, this approach finds optimal or near-optimal solutions in under 60 minutes. This results in a significant reduction in planning time for the schedulers.
We have also developed a validation layer that detects infeasibilities and conflicts before optimization. This component has proven essential in our deployed system, where schedulers need actionable feedback to adjust inputs when constraints cannot be satisfied.
In the talk, we present the MIP formulation, discuss why standard decomposition strategies fail for this problem class, describe the validation layer, and show computational results for schools with 300-1250 Students and practical experiences from our deployment at Finnish schools.

Keywords: Student sectioning · Course timetabling · Column generation · Large neighborhood search · Mixed-integer programming.

Instructor Scheduling in UniTime

Tomáš Müller

Purdue University, West Lafayette, Indiana, USA
muller@unitime.org

1 Introduction

Instructor scheduling is the problem of assigning instructors to classes subject to various constraints, such as instructor availability, qualifications, maximal load, teaching preferences, etc. While this problem can be included in the course timetabling component (instructors are assigned together with class times and rooms), instructors are usually assigned before or after the course timetabling [10]. Typically, professors or lecturers are already known when the course timetable is being built, and avoiding instructor clashes is a constraint commonly present in course timetabling problems [9]. On the other hand, teaching assistants and additional instructors are assigned after the class schedule has been published.

UniTime[1] is an open-source tool for university course [9] and examination [4] time-tabling, student [7] and instructor scheduling, and event management. While we have published several papers on various components over the years, the problem of assigning instructors has been neglected. It should also be noted that real-world timetabling data collected using UniTime has been used for the ITC 2019 competition [8], and a version of the UniTime solver has been used by the author in the ITC 2007 competition [3], where it was among the finalists in all three tracks and won two of them. While the same solver is used for instructor scheduling as for the other components, the problem model differs, with some interesting and challenging constraints.

The work on this module has been motivated by the teaching assistant assignments at Purdue University. Purdue University is a large public university with over 40 thousand students at its main campus in West Lafayette, Indiana, USA. Many departments, such as Mathematics, Chemistry, or Biology, offer large service courses attended by hundreds of students each semester. These courses usually consist of a small number of large lectures taught by faculty members and dozens of small lab sections, each manned by a teaching assistant. At Purdue, the course timetable is usually set about six months in advance, but teaching assistants are only assigned a few weeks before classes start, ensuring they can attend the classes they need for their studies. Additional personnel may need to be hired to cover the teaching load.

Time-wise, instructor scheduling occurs after course timetabling is complete, but it may overlap with student scheduling. This means that any instructor assignments are also considered during student scheduling, preventing a student

[1] https://www.unitime.org

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

who is also a teaching assistant from making a registration change that would conflict with their teaching schedule. However, in theory, the individual modules can be used in any order, as instructor assignments are also checked for conflicts during course timetabling. This provides a lot of flexibility, especially when dealing with disruptions that occur later in the scheduling process.

In this abstract, we describe the university instructor scheduling problem (UISP) as modeled in UniTime and briefly explain the algorithm used to solve it. For more details about the module, please refer to the UniTime documentation[2].

2 Instructor Scheduling Problem

The UISP consists of teaching requests and instructors. While these are mostly teaching assistants at Purdue, other teaching staff, such as professors and lecturers, can be included in the problem if desired. Each **instructor** has

- a **maximal teaching load**, typically expressed in the number of hours
- **teaching preference**, providing an optional penalization of an instructor if they are assigned a teaching request; this is used, e.g., to penalize outside personnel that would need to be hired, and would represent an additional cost for the department
- a **list of attributes** that can present various qualifications, skills, certifications, or other department-defined criteria that the instructor meets
- **time preferences** that allow an instructor to prohibit certain times and/or prefer/discourage some other times; time preferences are provided for individual intervals during the week, e.g., preferred Mondays 8 AM - noon, prohibited Fridays
- **course preferences** that allow an instructor to require, prefer, discourage, or prohibit certain courses
- **distribution preferences** that allow an instructor to provide preferences or requirements about the distribution of their class assignments, such as preferring back-to-back classes or requiring the same room for all their teaching assignments

As with other modules, UniTime uses a seven-level scale for preferences and requirements: **prohibited** and **required** preference levels are treated as hard constraints and must always be satisfied by the solver. All other preference levels have associated penalties: **strongly discouraged** (+4), **discouraged** (+1), **neutral** (0), **preferred** (-1), and **strongly preferred** (-4).

A **teaching request** represents a class that needs an instructor assigned to it. A single teaching request may contain multiple classes, as a course may have lab-recitation pairs that require the same instructor, or the instructor may need to be present at the corresponding lecture as well, which is common to multiple teaching requests. A teaching request has

- a **teaching load** expressing how many hours the instructor will need

[2] https://help.unitime.org/manuals/instructor-scheduling

- **one or more classes** of a course that the instructor would teach; along with their scheduled times and rooms, if already timetabled
- optional **common class or classes** of the course during which the instructor also needs to be available; typically, a part of the course common to multiple teaching requests, such as a lecture
- **attribute preferences** that include requirements and preferences for various attributes, such as qualifications, skills, certifications, etc.
- **instructor preferences** that allow for individual instructors to be preferred, discouraged, prohibited, or required
- **same-course** and **same-common** preference

Instructor attributes and teaching request attribute preferences allow modeling a fairly complex relationship between classes that need an instructor assigned (teaching requests) and instructors available to teach those classes or courses. For example, certain qualifications may be required, while other skills or certifications are preferred or discouraged. The design allows attributes to be grouped by type, and each type may specify whether its attributes are conjunctive or disjunctive. If multiple attributes of the same type are required on a teaching request, an instructor may need to have all the required attributes (conjunctive) or just one of them (disjunctive). This allows complete customization of attribute types and individual attributes.

The same-course and the same-common preferences arise from using the UISP primarily to assign teaching assistants. Typically, a teaching assistant has two, three, or four teaching assignments that must be from the same course and are preferred to share a common part (e.g., are associated with the same lecture).

A solution to a UISP is a set of teaching assignments, where each teaching request is assigned to one instructor. While partial solutions are permitted (some teaching requests may be left unassigned), all hard constraints must always be satisfied. Moreover, the aim is to minimize the weighted sum of the various optimization criteria (soft constraints/penalizations).

The problem has the following hard constraints that must always be satisfied

- The **teaching load** of an instructor cannot exceed the instructor's maximal load.
- The instructor **must be available** during their teaching assignment; i.e., there is no overlap with a prohibited time or any class they attend.
- An instructor's **teaching assignments cannot overlap** in time or share the same class, except for the common class.
- Instructors with a **prohibited teaching preference** cannot be used.
- Required and prohibited **attributes**, **instructor**, **course** and **distribution preferences** must be obeyed.
- Required and prohibited **same-course**, **same-common** must be obeyed. E.g., when an instructor has one teaching request that requires the same-course, all their assignments must be for the same course; a prohibited same-common would mean that no other assignment of the instructor shares any of the common classes of the request.

The optimization criteria (soft constraints) include the following criteria. Each represents a sum of the penalizations from negative (-4 for strongly preferred and -1 for preferred) to positive (+1 for discouraged and +4 for strongly discouraged) applied based on whether the preferred/discouraged time, course, instructor, etc., is used:

- **teaching preference**: applies to each teaching assignment of the instructor
- **time preference**: penalization of the worst preference overlapping with the teaching assignment
- **course preference**: penalization of a course preference set on an instructor
- **instructor preference**: penalization of an instructor preference set on a teaching request
- **attribute preference**: sum of penalizations of all attribute preferences that match the instructor attributes
- **distribution preference**: penalization for each violation of a distribution preference
- **unused instructor load** for instructors with one or more assignments; that is, the difference between the maximal load of the instructors and the sum of the loads of the assigned teaching assignments

Each of the above criteria has a weight in the solver configuration. The aim is to find a complete solution that minimizes the objective function, defined as a weighted sum of the above criteria.

3 Algorithm

The presented UISP is modeled as a Constraint Satisfaction Problem, where teaching requests are modeled as variables to be assigned. The domain of a teaching request is a list of instructors who can teach the request, meaning they satisfy the required/prohibited time, course, attribute, and instructor preferences and do not overlap with any class the instructor is attending as a student. This reduces the hard constraints that need to be checked during the search to the maximum load, teaching assignment overlaps, required/prohibited distributions, and required/prohibited same-course and same-common settings among the teaching assignments of the same instructor.

As with the other UniTime modules, the solver is implemented using the CPSolver library [2], sharing the common primitives such as variables, values, constraints, optimization criteria, search algorithms, and general heuristics with the other UniTime problems, and implementing only the problem-specific components. The CPSolver source code is available at GitHub[3], including the instructor scheduling solver.

The search algorithm starts with the iterative forward search (IFS) [6], in which, at each iteration, one unassigned variable is selected and assigned. This can lead to the unassignment of conflicting assignments. The conflict-based

[3] https://github.com/UniTime/cpsolver

statistics (CBS) [5] are used to prevent the algorithm from getting stuck in a local optimum and to provide the user with information when a complete solution cannot be found. Once a complete solution is obtained, the great deluge (GD) [1] technique is used to optimize it. The bound is dynamically set based on the best-known solution and reset when it falls below a threshold. During this phase, only instructor swaps and reassignments that do not violate any hard constraints can be selected, forcing the solver to move from one complete solution to another. Besides simple swaps and reassignments, the search employs a limited-depth branch-and-bound.

4 Conclusion

Among all the UniTime modules, the instructor scheduling module usually receives the least attention. This is because the individual problems are typically very small, involving only teaching assistants and/or other instructors within a single department.

It is possible that instructor scheduling will be integrated with the course timetabling module in the future, perhaps in a similar way to how student sectioning is combined with class assignments to time and space [9, 8].

References

1. Dueck, G.: New optimization heuristics: The great deluge algorithm and the record-to record travel. Journal of Computational Physics **104**, 86–92 (1993)
2. Müller, T.: University course timetabling: Solver evolution. In: Practice and Theory of Automated Timetabling 2016 Proceedings. pp. 263—-282 (2016)
3. Müller, T.: Itc2007 solver description: a hybrid approach. Annals of Operations Research **127**, 429–446 (2009). https://doi.org/10.1007/s10479-009-0644-y
4. Müller, T.: Real-life examination timetabling. Journal of Scheduling **19**, 257–270 (2016). https://doi.org/10.1007/s10479-014-1643-1
5. Müller, T., Barták, R., Rudová, H.: Conflict-based statistics. In: Gottlieb, J., Silva, D.L., Musliu, N., Soubeiga, E. (eds.) EU/ME Workshop on Design and Evaluation of Advanced Hybrid Meta-Heuristics. University of Nottingham (2004)
6. Muller, T., Barták, R., Rudová, H.: Iterative forward search: Combining local search with maintaining arc consistency and a conflict-based statistics. In: LSCS'04 — 1st International Workshop on Local Search Techniques in Constraint Satisfaction (2004)
7. Müller, T., Murray, K.: Comprehensive approach to student sectioning. Annals of Operations Research **181**, 249–269 (2010)
8. Müller, T., Rudová, H., Müllerová, Z.: Real-world university course timetabling at the International Timetabling Competition 2019. Journal of Scheduling **28**, 247–267 (2025). https://doi.org/10.1007/s10951-023-00801-w
9. Rudová, H., Müller, T., Murray, K.: Complex university course timetabling. Journal of Scheduling **14**(2), 187–207 (2011)
10. Xue, G., Offodile, O.F., Razavi, R., Kwak, D.H., Benitez, J.: Addressing staffing challenges through improved planning: Demand-driven course schedule planning and instructor assignment in higher education. Decision Support Systems **187** (2024). https://doi.org/10.1016/j.dss.2024.114345

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Scheduling with No-Wait Decisions in Heavily Loaded TSN Networks*

Jesper Vines[1,2][0000−0002−1228−9759], Hannes Uppman[2][0000−0003−0003−9127], Mikaela Lindberg[2], and Elina Rönnberg[1][0000−0002−2081−2888]

[1] Linköping University, Sweden
jesper.vines@liu.se, elina.ronnberg@liu.se
[2] Saab Aeronautics, Sweden

Keywords: Logic-based Benders Decomposition · Scheduling · Time Sensitive Networking · Avionics

1 Introduction

Time Sensitive Networking (TSN), standardized as IEEE 802.1Q, is a collection of standards that extend the functionality of Ethernet. This technology can be used to guarantee high reliability and bounded latency, which is crucial when working with a hard real-time system, like an avionics system (the electronic system in an aircraft). Our work concerns joint routing and scheduling of time-triggered messages in TSN networks, a problem which is known to be computationally challenging. In practice, routing is therefore often done before the scheduling of traffic. However, in heavily loaded networks, particular routing decisions can render the resulting scheduling problem infeasible, even if the complete problem has a solution. Vlk, Hanzálek and Tang [1] present a Logic-based Benders Decomposition (LBBD) approach for this problem, in which the master problem routes the traffic and the scheduling is handled by the subproblem. This captures the way this problem is often handled in practice, while still protecting against unfortunate routing decisions. This approach can, however, fail at solving larger problem instances, since the resulting scheduling subproblem can become intractable. We therefore propose an extension to this approach, where more decisions are taken in the master problem. This makes it possible to make more iterations and solve more challenging instances. One of our extensions, presented here, uses a technique that is often applied in practice to reduce the complexity of the scheduling: *no-wait scheduling*. As the name implies, no-wait scheduling means that messages are not allowed to wait in switches before being transferred. We incorporate this decision into the LBBD framework, so that we maintain our guarantees while simultaneously making the subproblem easier to solve. This extension allows us to solve a larger number of problem instances and is a first step towards designing a LBBD scheme that can solve even more challenging instances.

* This work was funded by Vinnova within the National Aeronautical Research Program (NFFP8), project 2024-01259.

2 Problem Formulation and Modelling

In this section we present the problem and the decomposed model (CP2), proposed in [1], which we refer to as the baseline model. The network is modeled as a directed graph $(\mathcal{V}, \mathcal{E})$, with the set of nodes $\mathcal{V}$ representing the end systems and switches, and the set of edges $\mathcal{E}$ representing the network links. All network connections allow for full-duplex communication, so $\mathcal{E}$ is symmetric.

TSN devices need to be time-synchronized, and we denote the maximal difference between two clocks in the network by δ. Each edge $(a, b) \in \mathcal{E}$ has a propagation delay δ_{ab}, while each node $a \in \mathcal{V}$ introduces a switching delay δ_a. The ports in TSN switches contain up to eight first-in-first-out queues. We denote the number of queues available for scheduled traffic in the egress port to link $(a, b) \in \mathcal{E}$ by q_{ab} and, for convenience, the set of indices $\{1, \ldots, q_{ab}\}$ by $\mathcal{Q}_{ab}$.

A *stream* is a periodic data transfer from a source end system, known as a *talker*, to a destination end system, known as a *listener*. Let $\mathcal{S}$ be the set of all streams. For each stream $i \in \mathcal{S}$, we denote the talker by t_i, the listener by l_i and the time it takes to transfer stream i over a link $(a, b) \in \mathcal{E}$ by L_{iab}. The period of stream $i \in \mathcal{S}$ is denoted by T_i, and the hyper period T is defined as the least common multiple of all stream periods. It follows that the system is periodic with period T, and that it is sufficient to construct a schedule of this length. During an interval of length T, stream $i \in \mathcal{S}$ must occur $k = T/T_i$ times. These occurrences are referred to as stream *instances*. Let $\mathcal{K}_i = \{0, 1, \ldots, T/T_i - 1\}$ denote the set of instance indices for stream $i \in \mathcal{S}$. For convenience, we also use $\mathcal{K}_i^+ = \mathcal{K}_i \setminus \{0\}$. For the first instance of stream $i \in \mathcal{S}$, the earliest time a talker can start transferring data is known as the stream's *release time*, denoted by r_i, and the latest time the listener must obtain the data is known as its *deadline*, denoted by d_i. It is assumed that $0 \leq r_i \leq d_i \leq T_i, i \in \mathcal{S}$.

The model is decomposed into a MIP master problem, aiming to find a shortest path routing, and a feasibility CP scheduling subproblem. In the master problem, the binary variable r_{iab} is used to indicate if stream $i \in \mathcal{S}$ is routed over edge $(a, b) \in \mathcal{E}$ or not. Correct routing is enforced by the constraints

$$\sum_{b \in \delta^+(t_i)} r_{it_ib} = \sum_{a \in \delta^-(l_i)} r_{ial_i} = 1 \qquad i \in \mathcal{S}, \tag{1}$$

$$\sum_{b \in \delta^-(t_i)} r_{it_ib} = \sum_{a \in \delta^+(l_i)} r_{ibl_i} = 0 \qquad i \in \mathcal{S}, \tag{2}$$

$$\sum_{a \in \delta^-(b)} r_{iab} = \sum_{c \in \delta^+(b)} r_{ibc} \leq 1 \qquad i \in \mathcal{S}, b \in \mathcal{V} : b \neq t_i, b \neq l_i, \tag{3}$$

where $\delta^+(b)$ and $\delta^-(b)$ are the sets of outgoing and incoming edges from $b \in \mathcal{V}$. Later iterations of the master problem also include no-good cuts of the form

$$\sum_{(i,a,b) \in \mathcal{C}} r_{iab} \leq |\mathcal{C}| - 1. \tag{4}$$

Here $\mathcal{C}$ is a minimal (or close to minimal) subset of $\mathcal{S} \times \mathcal{V} \times \mathcal{V}$ such that no solution to the full problem exists where $r_{iab} = 1$ for every $(i, a, b) \in \mathcal{C}$. We find such

sets $\mathcal{C}$ by computation of conflicts (infeasible submodels) for the earlier, infeasible subproblems. We denote the value of the variable r_{iab} in the latest master problem solution by $\hat{r}_{iab}$. For every stream $i \in \mathcal{S}$, we define the subgraph $(\mathcal{V}_i, \mathcal{E}_i)$ of $(\mathcal{V}, \mathcal{E})$, where $\mathcal{E}_i = \{(a,b) \in \mathcal{E} : \hat{r}_{iab} = 1\}$ and $\mathcal{V}_i = \{a \in \mathcal{V} : \exists b \text{ s.t. } (a,b) \in \mathcal{E}_i \text{ or } (b,a) \in \mathcal{E}_i\}$.

In the subproblem, interval variables are used to model when the streams occupy the links or queues in the network. The interval I_{iabk} indicates when instance $k \in \mathcal{K}_i$ of stream $i \in \mathcal{S}$ occupies link $(a,b) \in \mathcal{E}_i$. The interval Q_{ib} indicates when the first instance of stream $i \in \mathcal{S}$ occupies a queue in a port in switch $b \in \mathcal{V}_i$. We further introduce the optional interval variable $\bar{Q}_{ibkq}$, that can either be present or absent in a solution, which indicates when instance $k \in \mathcal{K}_i$ of stream $i \in \mathcal{S}$ occupies queue $q \in \mathcal{Q}_{bc}$. The connection between these variables is modeled by the Alternative and PresenceOf constraints (5)–(6), which ensure that exactly one of $\{\bar{Q}_{ib0q}, q \in \mathcal{Q}_{bc}\}$ is present, and shares start and end times with Q_{ib}, and that this queue is chosen for all instances of a stream. The NoOverlap constraints (7)–(8) ensures that the interval variables present in the constraint sets do not overlap.

$$\text{Alternative}(Q_{ib}, \{\bar{Q}_{ib0q} : q \in \mathcal{Q}_{bc}\}) \qquad i \in \mathcal{S}, (b,c) \in \mathcal{E}_i, \tag{5}$$

$$\text{PresenceOf}(\bar{Q}_{ibkq}) = \text{PresenceOf}(\bar{Q}_{ib(k-1)q}) \qquad i \in \mathcal{S}, (b,c) \in \mathcal{E}_i, q \in \mathcal{Q}_{bc}, k \in \mathcal{K}_i^+ : b \neq t_i, \tag{6}$$

$$\text{NoOverlap}(\{I_{iabk} : i \in \mathcal{S}, k \in \mathcal{K}_i \text{ s.t. } (a,b) \in \mathcal{E}_i\}) \qquad (a,b) \in \mathcal{E}, \tag{7}$$

$$\text{NoOverlap}(\{\bar{Q}_{ibkq} : i \in \mathcal{S}, k \in \mathcal{K}_i \text{ s.t. } (b,c) \in \mathcal{E}_i\})\,(b,c) \in \mathcal{E}, q \in \mathcal{Q}_{bc}. \tag{8}$$

The scheduling constraints (9)–(13) in the CP subproblem enforce release times, deadlines, path ordering, and zero latency variation (*jitter*). Since the transmission interval over a link directly influences when the stream is placed in the following queue, the relations between the queue and link occupation intervals (14)–(17) are also enforced. For every stream $i \in \mathcal{S}$, the constraints are

$$\text{start}(I_{iab0}) \geq r_i \qquad (a,b) \in \mathcal{E}_i, \tag{9}$$

$$\text{end}(I_{iab0}) \leq d_i \qquad (a,b) \in \mathcal{E}_i, \tag{10}$$

$$\text{start}(I_{ibc0}) \geq \text{end}(I_{iab0}) + \delta_{ab} + \delta_b + \delta \qquad (a,b),(b,c) \in \mathcal{E}_i, \tag{11}$$

$$\text{start}(I_{iabk}) = \text{start}(I_{iab(k-1)}) + T_i \qquad (a,b) \in \mathcal{E}_i, k \in \mathcal{K}_i^+, \tag{12}$$

$$\text{size}(I_{iabk}) = L_{iab} \qquad (a,b) \in \mathcal{E}_i, k \in \mathcal{K}_i, \tag{13}$$

$$\text{start}(Q_{ib}) = \text{end}(I_{iab0}) + \delta_{ab} - \delta \qquad (a,b) \in \mathcal{E}_i : b \neq l_i, \tag{14}$$

$$\text{end}(Q_{ib}) = \text{start}(I_{ibc0}) \qquad (b,c) \in \mathcal{E}_i : b \neq t_i, \tag{15}$$

$$\text{start}(\bar{Q}_{ibkq}) = \text{start}(\bar{Q}_{ib(k-1)q}) + T_i \qquad (b,c) \in \mathcal{E}_i, q \in \mathcal{Q}_{bc}, k \in \mathcal{K}_i^+ : b \neq t_i, \tag{16}$$

$$\text{end}(\bar{Q}_{ibkq}) = \text{end}(\bar{Q}_{ib(k-1)q}) + T_i \qquad (b,c) \in \mathcal{E}_i, q \in \mathcal{Q}_{bc}, k \in \mathcal{K}_i^+ : b \neq t_i. \tag{17}$$

3 Further Developments of the Decomposition Scheme

The master problem is often easy to solve, since the networks are relatively small. However, solving a complete scheduling problem in the subproblem can become

challenging for larger problem instances, e.g. with a greater number of streams. This problem is discussed briefly in [1], where it is noted that the method often fails to complete more than a single LBBD iteration—either the subproblem is solved in the first iteration, or the solver times out. This can lead to the problem mentioned in the introduction, where an unfortunate initial routing decision can make it very hard to determine feasibility of the model. In order to make larger problem instances benefit from the LBBD framework, the subproblem needs to be easier to solve.

One way of achieving this is by taking additional decisions in the master problem, to impose greater restrictions on the subproblem. One such decision, that significantly impacts the tractability of the subproblem, is the no-wait decision. For each $i \in \mathcal{S}$, we add the binary decision variable n_i to the master model, indicating if stream i is not allowed to wait in the queue of any switch it is routed through. The objective function tries to maximize the use of no-wait, as well as minimize the path length. We denote the value of n_i in the last master iteration by $\hat{n}_i$, and the constraint

$$\text{start}(I_{ibc0}) = \text{end}(I_{iab0}) + \delta_b + \delta \qquad i \in \mathcal{S} : \hat{n}_i = 1, \tag{18}$$

is added to the subproblem. The no-good cuts take the shape

$$\sum_{(i,a,b)\in\mathcal{C}_r} r_{iab} + \sum_{i\in\mathcal{C}_n} n_i \leq |\mathcal{C}_r| + |\mathcal{C}_n| - 1, \tag{19}$$

where $\mathcal{C}_r$ and $\mathcal{C}_n$ are small sets such that no solution to the full problem exists where $r_{iab} = 1$ for $(i, a, b) \in \mathcal{C}_r$ and $n_i = 1$ for $i \in \mathcal{C}_n$. These sets are again derived from earlier, infeasible subproblems.

For these no-wait streams we only take a single scheduling decision, the send time from the talker, instead of needing a scheduling decision per link. This makes finding a solution for the subproblem, or determining infeasibility and generating cuts, easier. To evaluate our extended approach with the no-wait decisions in the master problem, we have made preliminary computational experiments using a set of 150 instances for larger networks (96 nodes, 220 edges and between 30 and 900 streams). We compare the results from the baseline model from [1] to our extended approach. For the former, 65 instances are solved and for the latter, 72 instances are solved. We can also see that, in the instances that are not solved within the relatively short timeout, we manage to increase the number of LBBD iterations significantly. This means a larger amount of routing options for the streams are evaluated, which increases the chances of finding a solution to the original problem.

Ongoing and future work include further strengthening of the master problem and development of additional acceleration strategies.

References

1. Vlk, M., Hanzálek, Z., Tang, S.: Constraint programming approaches to joint routing and scheduling in time-sensitive networks. Computers & Industrial Engineering **157**, 107317 (2021)

A Demonstration of an Integrated Project Management and Demo Scheduling Platform

Neel Gaglani* and Ender Özcan

School of Computer Science, University of Nottingham, Nottingham, UK
neel148g@gmail.com, ender.ozcan@nottingham.ac.uk

Abstract. This paper presents an integrated software that automate the end-to-end coordination of undergraduate projects, including panel assignment, administrative data processing, conflict detection, and demo scheduling. The system combines a modern web-based architecture with tailored heuristic and optimisation methods to address the complex resource allocation constraints typically managed manually. A key component of the platform is the demonstration timetabling engine, capable of generating feasible, balanced schedules under staff, room, and time constraints. This work illustrates the proposed system's easy-to-use interactive interface, and compares the performance of the two hyper-heuristic approaches tested on realistic problem instances. Together, these components demonstrate how heuristic optimisation and workflow automation can significantly reduce administrative overhead while improving fairness and reliability in large-scale academic scheduling.

Keywords: Timetabling · Great Deluge · Simulated Annealing · Metaheuristic · Hyper-heuristics

1 Introduction

Coordinating undergraduate final year projects involves multiple interconnected processes, including project allocation, panel assignment, file preparation, conflict detection, and the generation of a demonstration timetable. Manual spreadsheet-based methods are time-consuming and prone to human error, especially as student numbers grow. Automated timetabling is well-recognised as a highly constrained NP-hard problem [8], and educational scheduling has been extensively studied in the literature [4, 2].

This paper presents a unified software platform that automates the end-to-end workflow of the management of final year projects. The system was designed around a web-based architecture and integrates deterministic administrative tools with (meta)heuristic optimisation components. A key feature and focus of this demonstration is the automated generation of demonstration schedules; an NP-hard timetabling problem that must consider staff availability, break windows, capacity limits, and workload fairness.

* Corresponding author.

We showcase the system interface, architectural components, and search behaviour using two tuned selection hyper-heuristics [6] using the same reinforcement learning heuristic selection methods and different move acceptance methods: *Simulated Annealing* (SA) [12] and *Great Deluge* (GD) [3]. We will illustrate how the tool dynamically generates, evaluates, and updates schedules, including support for manual overrides, staff substitutions, and instantaneous analytics.

The remainder of this paper is structured as follows. Section 2 describes the overall system architecture, highlighting the modular design of the frontend, backend, and database components, and explaining how these interact to support real-time scheduling and administrative workflows. Section 3 focuses on the panel assignment while Section 4 on the demonstration scheduling challenge, detailing the constraints, objective function, and (meta)heuristic strategies used to generate feasible timetables. Finally, Section 5 concludes the paper with a summary of contributions.

2 System Architecture

The system follows a modular client–server architecture, similar to modern educational timetabling platforms, embedding component-based hyper-heuristic development support as in HyFlex [13]:

- **Frontend:** Implemented in React with Tailwind CSS, supporting interactive timetable views, availability management, and configuration of optimisation parameters.
- **Backend:** A Flask-based REST API performing data processing, panel assignment, conflict detection, and optimisation. The system follows clean separation of concerns as recommended in network software architecture design [7].
- **Database:** PostgreSQL with a relational schema adhering to 3NF principles [5], ensuring data integrity for Students, Staff, Projects, Panel Assignments, Schedules, and Availability.

The scheduling engine uses a data hydration phase to load constraint-relevant entities into memory for constant-time access. Neighbourhood operators, delta (incremental) evaluation, and reinforcement-learning–based operator selection draw on principles from selection hyper-heuristics [1, 11].

The system exposes its features through a demonstration interface, allowing users to inspect algorithm behaviour, manually adjust schedules, and visualise staff idle times and workload balances. It includes interactive timetable editing with real-time constraint validation, a staff availability editor, and reproducible multi-run optimisation with seed control. This architecture ensures that computationally expensive scheduling operations are efficient while maintaining strict usability for administrative staff.

3 Panel Assignment

Panel assignment presents its own challenge due to the requirement that each project must be allocated exactly three unique panel markers while ensuring workload fairness, feasibility, and adherence to academic constraints.

Problem Definition. Let P be the set of all projects, S the set of supervisors and M the set of panel members. For each project i, the solution assigns a set of three unique panel members $\{PM_{i,1}, PM_{i,2}, PM_{i,3}\} \subseteq M$. Each project also has a pre-determined main supervisor, denoted by S_i.

Hard Constraint. All three panel members must be unique, and the supervisor cannot be assigned as a panel member. Formally, the four assigned academic roles must be pairwise distinct:

$$|\{S_i, PM_{i,1}, PM_{i,2}, PM_{i,3}\}| = 4 \qquad \forall i \in P.$$

Soft Constraint 1 (Total Workload Balance). The total dissertation marking load W_j^{total} for staff member j should be distributed as evenly as possible. The target workload α represents the total number of panel seats ($3|P|$) divided by the number of available panel members ($|M|$), allowing a maximum deviation of 1:

$$\lfloor \alpha \rfloor \;\le\; W_j^{\text{total}} \;\le\; \lfloor \alpha \rfloor + 1, \qquad \text{where } \alpha = \frac{3|P|}{|M|}.$$

Soft Constraint 2 (Demo-Marker Balance). The demo workload of each panel member W_j^{demo} should be balanced, targeting an average value β. Using our approach, where the first panel member assigned to mark a project dissertation is also required to attend its demo, we ensure that workload balance is fully maintained across both dissertation marking and demo attendance for all panel members.

$$\lfloor \beta \rfloor \;\le\; W_j^{\text{demo}} \;\le\; \lfloor \beta \rfloor + 1, \qquad \text{where } \beta = \frac{|P|}{|M|}.$$

The primary issues arise from overlapping staff roles, many supervisors are also panel markers, which creates frequent conflicts where a supervisor may accidentally be assigned to mark their own student. Additional complications include uneven workload distribution across staff, the need to balance demo-marker assignments, and the high likelihood of conflict propagation when attempting naïve manual adjustments. To address this, the system implements a pattern-based construction heuristic, which first generates three permutations of available panel markers and cyclically maps these patterns onto the project list. This guarantees near-uniform workload distribution before any conflict resolution occurs. A subsequent greedy repair stage identifies supervisor conflicts and resolves them by performing constrained swaps that preserve uniqueness conditions and workload boundaries. This hybrid approach enables the system to generate valid, balanced panel assignments in deterministic time, eliminating the need for manual shuffling and preventing downstream scheduling conflicts.

4 Demonstration Scheduling

Demonstration scheduling is formulated as a multi-constraint optimisation problem comparable to classical university timetabling challenges. Each project requires assignment of a demo marker, supervisor, computer station, and time slot while satisfying the following strict hard constraints:

- No staff member (supervisor or marker) can be in two places simultaneously.
- Demonstrations occur across two fixed days (e.g., from 10:00–16:00) with a mandatory lunch break (e.g., from 12:00–13:00).
- Up to a predefined maximum number of parallel demonstrations/sessions may occur.
- Staff availability constraints must be honoured.

Soft constraints aim to improve fairness and convenience:

- Minimising idle gaps between sessions.
- Consolidating small workloads into a single day.
- Balancing large workloads evenly across both days.

We have developed a hyper-heuristic approach to tackle this problem. The design of the algorithmic components are presented in the following subsections.

4.1 Common Algorithmic Components

Solution Representation. The solution is represented as an array of project events:

$$\text{Events}_i = \{\text{DemoMarker}_i,\ \text{Day}_i,\ \text{TimeSlot}_i,\ \text{Computer}_i\}.$$

Objective Function. The objective (cost) function evaluates the quality $f(s)$ of a solution using a weighted sum of number of hard-constraint violations (H) and total penalties incurred due to all soft-constraint violations. Hard constraints, weighted by w_H, are prioritised significantly over soft constraints.

$$f(s) = (w_H \times H) + (\text{Penalty}_{\text{gap}} + \text{Penalty}_{\text{small}} + \text{Penalty}_{\text{heavy}}).$$

Three staff-centric targeted criteria are incorporated into the objective function to maximise staff convenience and improve the quality of generated demonstration timetables.

- *Gap Penalty.* To reduce idle time (gap_j) between consecutive demonstrations (in minutes) for any staff member j, gaps are penalised quadratically. Longer waiting periods incur disproportionately higher penalties:

$$\text{Penalty}_{\text{gap}} = (\sum_{\forall j} (\text{gap}_j)^2) \times w_{\text{gap}}.$$

– *Small Workload Consolidation.* For (c) staff members whose total number of assigned sessions is below a configurable threshold (initially set to 4), a fixed penalty of 2000 is applied if their sessions are split across both days. This encourages consolidating small workloads into a single day.

$$\text{Penalty}_{\text{small}} = c \times w_{\text{small}}.$$

– *Heavy Workload Balancing.* For staff members whose workloads exceed the threshold, the algorithm seeks to balance sessions across both days. A cubic penalty is imposed on imbalances, ensuring that heavier workloads are distributed equitably:

$$\text{Penalty}_{\text{heavy}} = (\sum_{\forall j}(|D_{j,1} - D_{j,2}|^3) \times w_{\text{heavy}}.$$

In our experiments, we have set $w_{\text{gap}} = 20$, $w_{\text{small}} = 2000$ and $w_{\text{heavy}} = 500$.

Initialisation. A construction heuristic has been employed to create the initial solution. This heuristic firstly groups projects by their assigned demo marker and sorts them by their load size. The algorithm then attempts to place these groups in consecutive blocks, splitting larger groups evenly across the two available demonstration days. Unplaced projects are then greedily assigned to any valid computers and timeslots available.

The search landscape is highly constrained and often dense, particularly when staff supervise many projects. The system implements two selection hyper-heuristics sharing a common objective function, heuristic selection method, neighbourhood operators, and delta (incremental) evaluation engine, but different move acceptance methods as outlined below.

4.2 Move Acceptance

Simulated Annealing (SA). Based on the classical formulation of Kirkpatrick et al. [12], SA uses geometric cooling (with $\alpha = 0.998$ and minimum temperature of 1.0). A dynamic reheating mechanism was integrated to combat the risk of "deep" local optima. If the algorithm failed to find a new global best solution for a continuous span of the maximum number of evaluations, the temperature would temporarily be multiplied by a boost factor. This would allow the heuristic to accept worse moves temporarily so that it can climb out of the local minimum trap before resuming its standard geometric cooling again.

Great Deluge (GD). Originally proposed as an alternative monotonic descent method [3], GD accepts solutions beneath a decaying water level. GD often performs better in compact search spaces with fewer deep local minima. The initial water level L_{initial} is set by evaluating the cost of the initially generated solution and applying a 5% buffer. This provides a brief exploratory phase before the acceptance boundary becomes restrictive:

$$L_{\text{initial}} = f(s_{\text{initial}}) \times 1.05.$$

The water level is then reduced linearly using a constant decay rate ΔL. To ensure that the algorithm transitions into a strict hill-climbing exploitation phase during the final portion of its execution, the decay rate is defined such that the water level reaches zero precisely at 90% of the total evaluation budget. Hence $\Delta L = L_{initial}/(\text{maximumNumberOfEvals} \times 0.9)$.

A hyper-heuristic terminates when the maximum number of evaluations is exceeded.

4.3 Low Level Heuristics

A selection hyper-heuristic controlling a set of perturbative operators typically consists of two key components: heuristic selection and move acceptance methods. In this work, we have combined an online reinforcement-learning heuristic selection mechanism, similar to multi-armed bandit strategies studied in hyper-heuristics [10], with Simulated Annealing and Great Deluge move acceptance methods as described above.

Four different perturbative neighbourhood operators are implemented as low-level heuristics and the selected operator is applied to the incumbent solution at each iteration of the hyper-heuristic.

- **LLH1: Magnetic Move.** Selects a random project and attempts to move it to a timeslot immediately preceding or following another project involving the same staff member.
- **LLH2: Repair Gap.** Identifies the staff member with the current largest idle-time gap and applies a magnetic move to reduce that gap.
- **LLH3: Consolidate Day.** Locates a staff member whose total workload falls below the configurable threshold and attempts to migrate all of their assigned projects into a single day.
- **LLH4: Repair Heavy Balance.** Detects a staff member with an overloaded and unbalanced schedule (above the threshold) and shifts one of their projects from the heavier day to the lighter day.

4.4 Heuristic Selection Method

The designed hyper-heuristics employ an online reinforcement learning mechanism to dynamically learn which operators are most effective at different stages of the search. To keep the computational overhead low, the search process is divided into discrete learning segments, or *epochs*, each consisting of 50 iterations. A segment size of 50 offers a suitable balance: it provides enough samples to measure operator performance while still reacting quickly to changes in the fitness landscape. During each epoch, the algorithm records the number of invocations of each operator and assigns a stratified reward based on the quality of the resulting move. At the end of each 50-iteration epoch, the average performance p_{avg} of each operator is computed by dividing its accumulated reward by

its number of applications. The selection probability weight for the next epoch is then updated using an exponential moving average:

$$w_{\text{new}} = (w_{\text{old}} \times (1 - \gamma)) + (p_{\text{avg}} \times \gamma),$$

where γ is the reaction factor, set to 0.5. This balance between historical memory (w_{old}) and immediate feedback (p_{avg}) allows the algorithm to shift operator preference as the nature of the timetabling bottleneck evolves. To avoid premature convergence and prevent any single operator from dominating the search, a hard lower bound of 0.05 is imposed on all weights. This ensures continued exploration across the solution space.

4.5 Performance Comparison of Hyper-heuristics Using Simulated Annealing and Great Deluge

To compare the performance of hyper-heuristics, random realistic problem instances with varying characteristics in terms of size and conflict density were generated based on past datasets[1]. The problem instances range from 100 to 180 projects and represent different staff–project interlock patterns. The characteristics of each problem instance from the demo scheduling aspect are provided in Table 1.

Table 1. Structural characteristics of the five randomly generated datasets (problem instances).

ID	$\lvert S\rvert$	$\lvert M\rvert$	$\lvert S \cap M\rvert$	$\lvert P\rvert$	**Density**
1	60	40	20	180	21.6%
2	50	30	24	125	31.4%
3	42	20	6	160	37.3%
4	30	20	8	100	41.6%
5	49	35	21	150	42.8%

Each hyper-heuristic algorithm with a different move acceptance method was run ten times using the same computational budget of 7,000 maximum evaluations. Across all problem instances, the average computation time per run was 19.71 seconds for Simulated Annealing and 20.56 seconds for Great Deluge, with individual instance averages ranging from 14.11 to 24.89 seconds. This rapid execution further validates the system's suitability for practical, real-time administrative usage.

Table 2 summarises the mean penalty cost (lower is better), based on the shared multi-component objective function, and Figure 1 provides an overview of box plot illustrations for the SA and GD-based hyper-heuristics, run on each instance.

[1] link to the github repository: https://github.com/neelg148/TestingDatasets

Table 2. Performance comparison of SA and GD-based hyper-heuristics for each of the five datasets based on mean cost and best cost, obtained over the ten runs.

ID	SA Mean Cost	GD Mean Cost	SA Best Cost	GD Best Cost
1	**1,593,100**	1,616,300	**1,022,500**	1,228,000
2	1,080,850	**711,650**	**472,500**	542,500
3	**2,139,950**	2,827,800	**1,800,000**	2,315,500
4	1,032,550	**463,950**	693,000	**225,500**
5	1,708,750	**1,597,400**	1,219,000	**1,140,000**

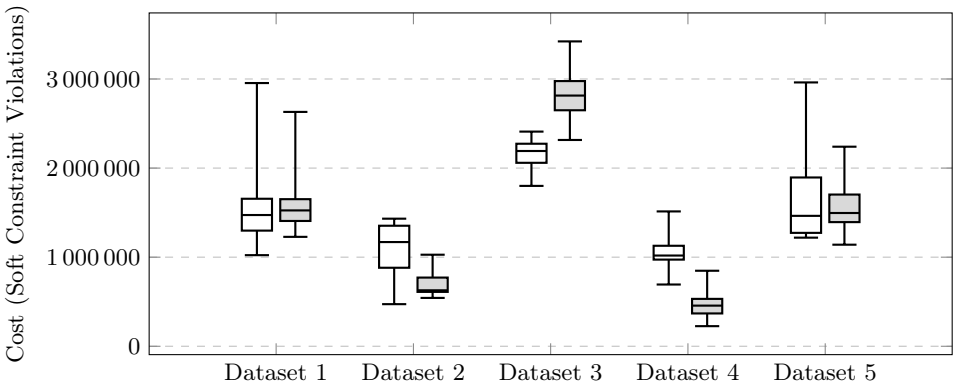


Fig. 1. Box-plot illustrations comparing the distribution of final penalty costs from 10 runs across five problem datasets. **White boxes** represent Simulated Annealing (SA) and **grey boxes** represent Great Deluge (GD).

This study illustrates the following observations, supported by prior findings on the behaviour of local search heuristics [9].

- **All hard constraints are satisfied**, as mean costs in Table 2 indicate mean soft constraint violations, averaged over the 10 runs for each instance.
- **SA-based hyper-heuristic obtains the best solution in 10 runs for 3 problem instances** (Datasets 1, 2, 3), performing slightly better than the GD-based hyper-heuristic as illustrated in Figure 1.
- **GD-based hyper-heuristic excels on smaller, compact search spaces** (Datasets 2 and 4), providing statistically significant improvements due to its deterministic descent based on their average performances.
- **SA-based hyper-heuristic outperforms GD-based hyper-heuristic on dense, high-interaction datasets** (Dataset 3), due to its ability to escape deep local minima via probabilistic acceptance and reheating based on their average performances.
- **Neither algorithm universally dominates**, reinforcing the system's design decision to include both methods and allow the user to choose based on dataset characteristics based on average performance comparison.

5 Conclusion

This paper presents an integrated software platform [2] that automates the full project coordination workflow, with emphasis on solving the complex demonstration scheduling problem using two high-performing hyper-heuristics. Through an interactive interface (see Figure 2), users can explore the scheduling engine, compare algorithmic behaviours, and observe real-time optimisation outcomes. The system demonstrates how heuristic optimisation, careful data engineering, and user-centered design can be combined to improve high-stakes academic administrative processes.

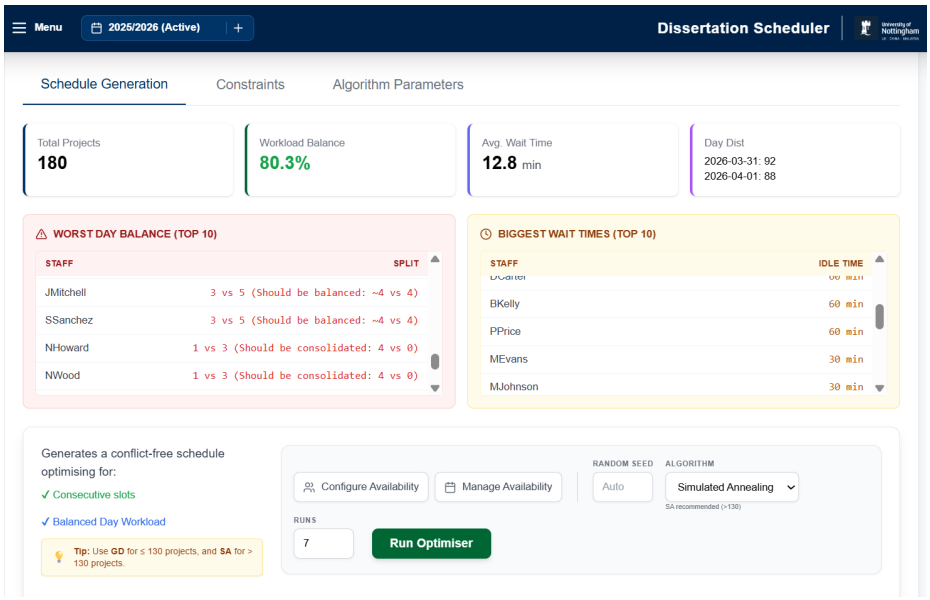


Fig. 2. The interactive user interface of the demonstration scheduling platform.

The system's usability and core functionality were thoroughly tested and validated through User Acceptance Testing, and the software is scheduled for full real-world deployment in the upcoming academic year. It will be installed locally on the module convenor's hardware to manage the live dissertation cohort.

During the design phase, exact optimisation techniques such as Integer Linear Programming (ILP) were also considered. However, these approaches were ultimately deemed impractical due to their susceptibility to combinatorial explosion when applied to the full-scale NP-hard scheduling problem. Consequently, their exploration has been deferred as a direction for future work. A further avenue for

[2] Source code of the implementation: https://github.com/neelg148/Dissertation-Management-Tool

future work is to investigate the application of algorithm selection techniques [14] to identify the most suitable algorithm based on instance-specific features of the demo scheduling problem.

References

1. Burke, E., Kendall, G., Soubeiga, E.: A survey of hyper-heuristics. In: *Handbook of Metaheuristics* (2003)
2. Burke, E., Kendall, G., Soubeiga, E.: A tabu-search hyperheuristic for timetabling. *Artificial Intelligence Review* **9**, 451–457 (2004)
3. Bykov, Y.: The great deluge algorithm and its application to the course timetabling problem. *PATAT* (2003)
4. Ceschia, S., Gaspero, L., Schaerf, A.: Educational timetabling: Problems, benchmarks, and state-of-the-art results. *Artificial Intelligence Review* **(308)**1 (2023)
5. Connolly, T., Begg, C.: *Database Systems: A Practical Approach*. Pearson (2014)
6. Drake, J.H., Kheiri, A., Özcan, E., Burke, E.K.: Recent advances in selection hyperheuristics. Eur. J. Oper. Res. **285**(2), 405–428 (2020).
7. Fielding, R.: Architectural styles and the design of network-based software architectures. PhD Thesis, UC Irvine (2000)
8. Garey, M., Johnson, D.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman (1979)
9. Jackson, W., Özcan, E., John, R.: Move Acceptance in Local Search Metaheuristics for Cross-domain Search. *Expert Systems with Applications* **109** 131–151,(2018)
10. Jackson, W., Özcan, E., John, R.: Tuning simulated annealing for cross-domain search. *Proc. of IEEE Congress on Evolutionary Computation (CEC)*, 1055–1062 (2017)
11. Kheiri, A., Özcan, E.: High school timetabling worldwide using selection hyperheuristics. *Expert Systems with Applications* **42**(13), 5463–5471 (2015)
12. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by simulated annealing. *Science* (1983)
13. Ochoa, G., Hyde, M., Curtois, T., Vörnkorg, J., Burke, E.K.: HyFlex: A benchmark framework for cross-domain heuristic search. In: Hao, J.-K., Middendorf, M. (eds.) EvoCOP 2012, LNCS **7245**, 136–147. Springer, Berlin, Heidelberg (2012)
14. Rice, J.R.: The algorithm selection problem. Adv. Comput. **15**, 65–118 (1976).

A Bandit-Based Reinforcement Learning Simulated Annealing for the Omnichannel Last-Mile Distribution

Vincent F. Yu[1], Renan S. Maglasang[1], Aldy Gunawan[2], Ching-Hsuan Lin[1], Kuan-Fu Chen[3], and Shih-Wei Lin[4]

[1] Department of Industrial Management, National Taiwan University of Science and Technology, Taipei 106, Taiwan, {vincent,renan,m10901001}@ntust.edu.tw
[2] School of Computing and Information Systems, Singapore Management University, Singapore, Singapore, aldygunawan@smu.edu.sg
[3] College of Intelligent Computing, Chang Gung University, Taoyuan 333, Taiwan, kfchen@mail.cgu.edu.tw
[4] Department of Information Management, Chang Gung University, Taoyuan 333, Taiwan, swlin@mail.cgu.edu.tw

Abstract. Omnichannel last-mile distribution requires vehicles to manage deliveries, returns, and inventory across multiple fulfillment sources. In omnichannel retail, a central warehouse replenishes stores, customer orders are fulfilled from stores or warehouses based on inventory, and returns may be collected during delivery routes. This study introduces the Vehicle Routing Problem in Omnichannel Distribution with Simultaneous Pickup and Delivery (VRPOSPD), covering delivery-only, pickup-only, and simultaneous pickup-and-delivery operations under store-only and hybrid store–depot fulfillment scenarios. A Bandit-Based Reinforcement Learning Simulated Annealing (BBRLSA) algorithm with adaptive neighborhood selection and reward-based learning is proposed to solve the problem. Computational results show that BBRLSA outperforms conventional simulated annealing in terms of solution quality, robustness, scalability, and CPU time particularly under hybrid fulfillment settings.

Keywords: Vehicle routing problem · Omnichannel last-mile distribution· Reinforcement learning · Simulated annealing.

1 Introduction

E-commerce growth has increased the complexity of last-mile omnichannel distribution beyond traditional systems [2]. Vehicles must handle deliveries, returns, and dynamic loads from stores or depots, creating routing–inventory interdependencies that challenge classical VRPs [4]. While pickup-and-delivery models provide a foundation [3], most assume paired requests [7] and overlook shared fulfillment nodes. Recent studies incorporate store-based fulfillment and returns [6], but routing and inventory decisions are still often treated separately.

We introduce the VRP in Omnichannel Distribution with Simultaneous Pickup and Delivery (VRPOSPD), involving a depot, stores, and customers with het-

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

erogeneous requests under pooled inventory. Two scenarios are studied: store-only (M1) and hybrid store–depot (M2). To solve this tightly coupled problem, we propose a Bandit-Based Reinforcement Learning Simulated Annealing (BBRLSA) framework that improves adaptive neighborhood selection using magnitude-based rewards, recent-biased learning, and ε-greedy exploration [5].

Building on the omnichannel last-mile distribution setting of Abdulkader et al. [1], where warehouses replenish stores and online orders are fulfilled based on inventory availability, the VRPOSPD extends the setting by incorporating customer returns and simultaneous pickup-and-delivery operations. Consider a network $\mathcal{N} = \{0\} \cup \mathcal{R} \cup \mathcal{C}$, where node 0 denotes a central depot, $\mathcal{R}$ the set of retail stores, and $\mathcal{C}$ the set of online customers. A homogeneous fleet of vehicles k is dispatched from the depot to carry out delivery and pickup operations within the same route. Routing decisions are whether vehicle k travels directly from node i to node j or not. Retail stores—and sometimes the depot—hold pooled inventories of product types $p \in \mathcal{P}$ and act as fulfillment nodes. Inventory is homogeneous within each type and not preassigned to customers; pickups represent replenishment rather than specific requests. Customers are divided into pickup-only $\mathcal{C}^{(1)}$, delivery-only $\mathcal{C}^{(2)}$, and simultaneous pickup-and-delivery $\mathcal{C}^{(3)}$ sets. Deliveries specify product type and quantity, while returns increase vehicle load. For each delivery customer, a fulfillment node f is chosen. Two scenarios are considered: Model M1 restricts fulfillment to retail stores ($f \in \mathcal{R}$), whereas Model M2 allows fulfillment from stores or the depot ($f \in \mathcal{R} \cup 0$). Inventory feasibility at fulfillment nodes and consistency between routing and sourcing decisions are enforced. Vehicles may replenish multiple times along a route, but each store can be visited at most once; when store s is visited, the vehicle collects the aggregated delivery quantities for all customers assigned to s and served later on that route. In both scenarios, routes start and end at the depot, each customer is visited exactly once, and flow conservation and fulfillment–customer precedence are maintained. The VRPOSPD aims to minimize total routing cost, measured by total vehicle travel distance, while satisfying routing, fulfillment, inventory, precedence, and capacity constraints.

2 The Proposed Algorithm

BBRLSA extends simulated annealing by incorporating a bandit-based adaptive operator selection mechanism. To generate neighboring solutions, BBRLSA uses three operators: insertion, swap, and inversion. Insertion relocates a randomly selected customer within or across routes, swap exchanges two customer nodes, and inversion reverses a subsequence within a route. Due to store replenishment and fulfillment precedence constraints in VRPOSPD, all generated solutions are checked for fulfillment consistency, store precedence, inventory feasibility, and vehicle capacity. Infeasible solutions are handled via penalization within the simulated annealing framework. During the search, one neighborhood operator is selected at each iteration using an ε-greedy reinforcement learning mechanism.

All operators are initially assigned equal probabilities, which are then updated adaptively based on historical rewards from improvements they generate.

The algorithm starts with an initial solution, which is treated as both the current solution and the best-known solution, together with an initial temperature. Each neighborhood operator is initialized with a performance estimate and usage count to facilitate adaptive learning throughout the search process. At each temperature level, a fixed number of neighborhood evaluations is performed. During each iteration, a neighborhood operator is selected using an ε-greedy policy. With probability 1-ε, the operator with the highest estimated performance is selected (exploitation), whereas with probability ε, a neighborhood operator is selected randomly (exploration). This mechanism balances search intensification and diversification while enabling the algorithm to progressively favor operators that historically produce higher-quality improvements.

The selected operator generates a candidate solution, which is evaluated using a penalized objective function. Acceptance follows the simulated annealing criterion: improving solutions are accepted deterministically, while worsening solutions may still be accepted with a probability that depends on the temperature and the magnitude of deterioration. This allows the algorithm to escape local optima. After generating a neighboring solution, a reward is assigned based on the relative improvement in the penalized objective function. Operators that achieve larger reductions receive higher rewards, while non-improving moves receive zero reward. Operator quality is updated via incremental averaging, allowing the algorithm to favor consistently effective operators while maintaining exploration through the ε-greedy mechanism. The temperature is reduced geometrically at each level, and the algorithm terminates after a fixed number of consecutive non-improving temperature levels. Overall, BBRLSA combines adaptive learning and simulated annealing to efficiently guide the search toward high-quality routing solutions.

3 Preliminary Results

Table 1 compares the performance of ESA/SA and BBRLSA on large-scale store-fulfillment (M1) and hybrid-fulfillment (M2) VRPOSPD instances. The instances are based on large VRPO instances [1], which include a single warehouse, customers, demand quantities, store stock levels per product, and the x–y coordinates of each node. These instances were modified to incorporate simultaneous pickup-and-delivery and return operations, with 10–25 stores and 25–150 customers. Customer demand is generated randomly as pickup-and-delivery, pickup-only, or delivery-only. Each scenario contains 60 instances classified into low-, medium-, and high-inventory levels. The Average Percent Gap ($\%Gap_{Ave}$) measures the solution quality relative to the best-known solution (BKS) values obtained from ten runs per instance:

$$\%Gap_{Ave} = \frac{1}{n}\sum_{i=1}^{n}\%Gap_i, \quad \%Gap_i = \frac{Obj_i^{ave} - Obj_i^{BKS}}{Obj_i^{BKS}} \times 100$$

The Success Rate represents the percentage of runs that exactly attain the BKS:

$$SR = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\left(Z_i^{best} = Z_i^{BKS}\right) \times 100$$

where n is the total number of instances, Z_i^{best} is the best objective value obtained by the algorithm for instance i, Z_i^{BKS} is the best-known solution value for instance i, and $\mathbb{I}(\cdot)$ is an indicator function equal to 1 if $Z_i^{best} = Z_i^{BKS}$ and 0 otherwise. Both SA and BBRLSA were implemented in Microsoft Visual C++ 2019 and executed on a desktop equipped with an Intel(R) Xeon(R) CPU E3-1245 v6 @ 3.70 GHz and 64 GB of RAM under Windows 11. The reported $\%TimeDiff$ measures the relative computational time difference between BBRLSA and SA under the same computing environment, where a negative value indicates that BBRLSA requires less computational time than SA.

Results show that M2 instances achieve lower average percentage gaps and higher success rates than M1 for both algorithms, suggesting that hybrid fulfillment increases routing flexibility. BBRLSA also shows stronger improvements over SA in M2. On average, BBRLSA reduces the gap by 0.50 percentage points in M2 but slightly worsens it by 0.05 in M1. Improvements in M1 are small and instance-dependent, whereas M2 shows more consistent gains, especially in low- and medium-inventory groups, with gap reductions of up to 0.5 percentage points and success rate increases up to 20%. In terms of CPU time, BBRLSA is 8.967% times faster than SA in M1 and 7.351% times faster than SA in M2 as indicated by the negative $\%TimeDiff$ values.

	%Gap-Ave		Success Rates (%)		CPU Time (BBRLSA, s)			
	SA	BBRLSA	SA	BBRLSA	Min	Max	Ave	%TimeDiff
M1 instances								
Overall Performance	**3.39**	**3.44**	**53.33**	**60.00**	**14.2**	**420.9**	**150.4**	**-8.967**
M1:01–20 instances (low)	5.12	5.35	50.00	55.00	13.6	424.3	153.5	-8.790
M1:21–40 instances (medium)	3.03	3.02	50.00	70.00	14.3	407.4	150.2	-8.934
M1:41–60 instances (high)	2.02	1.96	60.00	55.00	14.7	431.0	147.3	-9.176
M2 instances								
Overall Performance	**2.45**	**1.95**	**60.00**	**71.67**	**21.9**	**543.6**	**201.4**	**-7.351**
M2:01–20 instances (low)	2.29	1.65	60.00	80.00	17.9	551.7	198.4	-8.108
M2:21–40 instances (medium)	2.64	1.98	55.00	70.00	19.3	578.6	200.9	-7.132
M2:41–60 instances (high)	2.41	2.21	65.00	65.00	28.4	500.6	205.0	-6.813

Table 1. Performance comparison of SA and BBRLSA.

4 Conclusion

This paper introduces the Vehicle Routing Problem in Omnichannel Distribution with Simultaneous Pickup and Delivery (VRPOSPD), integrating routing, fulfillment, and loading decisions beyond classical models. We examine store-only and hybrid store–depot scenarios to assess sourcing flexibility, and propose a Bandit-Based Reinforcement Learning Simulated Annealing (BBRLSA) algorithm with adaptive operator selection. Results show BBRLSA outperforms standard SA in solution quality, robustness, and scalability, and CPU time especially in hybrid settings, with statistical tests confirming consistent improvements.

References

1. Abdulkader, M., Gajpal, Y., ElMekkawy, T.Y.: Vehicle routing problem in omnichannel retailing distribution systems. International Journal of Production Economics **196**, 43–55 (2018)
2. Arslan, B., Schrotenboer, A.H., Atan, Z.: Optimal fulfillment and replenishment for omnichannel retailers with standard shipping contracts. European Journal of Operational Research **323**(2), 642–656 (2025)
3. Berbeglia, G., Cordeau, J.F., Gribkovskaia, I., Laporte, G.: Static pickup and delivery problems: A classification scheme and survey. TOP **15**(1), 1–31 (2007)
4. Hübner, A.H., Kuhn, H., Wollenburg, J., Towers, N., Kotzab, H.: Last mile fulfilment and distribution in omni-channel grocery retailing: a strategic planning framework. International Journal of Retail Distribution Management **44**(3) (03 2016)
5. Kahalimoghadam, M., Thompson, R.G., Rajabifard, A.: Self-adaptive metaheuristic-based emissions reduction in a collaborative vehicle routing problem. Sustainable Cities and Society **110**, 105577 (2024)
6. Mandal, P., Basu, P., Saha, K.: Forays into omnichannel: An online retailer's strategies for managing product returns. European Journal of Operational Research **292**(2), 633–651 (2021)
7. Parragh, S.N., Doerner, K.F., Hartl, R.F.: A survey on pickup and delivery problems. Journal für Betriebswirtschaft **58**(1), 21–51 (2008)

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

New Insights into Local Search for Curriculum-Based Course Timetabling

Sara Ceschia, Francesca Da Ros, Luca Di Gaspero, and Andrea Schaerf✉

Intelligent Optimization Laboratory, University of Udine, Udine, Italy
andrea.schaerf@uniud.it

Abstract. Curriculum-Based Course Timetabling (CB-CTT) is a long-standing benchmark problem in educational timetabling, yet obtaining robust performance across diverse datasets under strict time limits remains challenging. In this work, we revisit the role of local search for CB-CTT and present a Simulated Annealing (SA) method that uses the LectureKick neighborhood, a generalized version of the classic Lecture-Move neighborhood. The method combines moves preserving feasibility, sampling biases for soft constraints, efficient delta evaluation, and a cut-off mechanism for the cooling schedule. The algorithm is tuned on a large artificial training set and evaluated on the International Timetabling Competition 2007 instances and additional benchmark datasets. Experimental evidence shows that, under the original ITC 2007 time limits, the proposed approach clearly outperforms previously published local search methods and remains competitive with the state-of-the-art methods reported in the literature. These results indicate that a careful redesign of neighborhoods and move-selection biases can still significantly improve local search performance on a mature benchmark such as CB-CTT.

Keywords: Course Timetabling · Simulated Annealing · Neighborhood Relations

1 Introduction

Curriculum-Based Course Timetabling (CB-CTT) is one of the most widely studied university timetabling formulations, largely due to its adoption in Track 3 of the Second International Timetabling Competition (ITC 2007) and the continued use of its benchmark datasets [6]. In CB-CTT, lectures must be assigned to periods and rooms over a weekly horizon while satisfying hard constraints on room occupancy, conflicts, and teacher availability, and minimizing violations of soft constraints related to room capacity, room stability, minimum working days, and isolated lectures [4,6].

Over the years, CB-CTT has been addressed using local search, mathematical programming, and hybrid approaches [2]. Under the strict time limits of ITC 2007, local search has historically played a central role, from the competition-winning hybrid solver by Müller [7] to later tabu search, simulated annealing, and hybrid large neighborhood approaches. Nonetheless, the large

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

number of neighborhoods proposed in the literature has made it difficult to identify which components are truly responsible for performance gains.

The aim of this work is to revisit local search for CB-CTT through a systematic consolidation of neighborhood ideas previously proposed in the literature. Our goal is to show that a careful redesign of move generation, neighborhood biases, and parameter tuning can substantially improve the effectiveness of a classic Simulated Annealing (SA) framework. SA was chosen based on both the strong performance reported in the literature and the extensive expertise developed by our research group through the solution of other timetabling problems. The resulting method is evaluated under the original ITC 2007 time limits and compared against strong local search baselines and state-of-the-art reference methods. All instances and best solutions are available on our website OptHub for inspection and future comparison.

2 Solution Method

The initial solution is constructed in three phases. First, a randomized construction procedure generates a complete assignment that satisfies all hard constraints except, possibly, curriculum and teacher conflicts. A short hill climbing phase based on lecture relocations and lecture swaps is then used to reach feasibility. A second hill climbing stage, still using the same neighborhoods, quickly improves the feasible solution before the main search begins.

The core of the method is an SA procedure operating on a neighborhood called LectureKick. The move extends Müller's LectureMove [7] in the following way: a lecture is moved to a target slot, and if that slot is occupied, the conflicting lecture is "kicked" to another admissible slot instead of being forced back to the original one (as in LectureMove). This generalization increases flexibility while preserving feasibility. Move generation is incremental and supported by auxiliary data structures storing admissible periods, conflicts per course-period pair, and room-related information, so that infeasible partial moves can be discarded early and delta costs can be evaluated efficiently.

To bias the search toward high-quality solutions, the neighborhood is not sampled uniformly. Four mutually exclusive biases target the soft constraints: one favors moves that improve minimum working days, one focuses on critical courses that are likely to affect isolated lectures, and two select lectures involved in room capacity and room stability violations. Two additional independent biases are used to avoid introducing new room capacity violations and to favor swap-like moves when beneficial. A preliminary screening phase showed that not all biases are equally relevant; in the final configuration, the most effective components have been retained and tuned. The final parameter configuration was obtained with `irace` on 996 artificial instances selected from the generator of De Coster et al. [3].

The SA schedule follows the classical Metropolis acceptance rule, combined with a geometric cooling schedule and a cut-off mechanism that accelerates the early stages of the search without reducing the total computational budget.

3 Experimental Results

First, we considered the `comp` dataset introduced at ITC 2007, following the competition protocol (single core and time limit). On the 19 non-trivial instances of this dataset, the proposed SA improves the best previously reported average result among local search methods on 18 instances and ties on the remaining one. The method achieves an average rank of 1.11, clearly outperforming the best local search solvers from the literature.When compared with the state-of-the-art methods [1,5] under the competition time limit, our SA obtains the best overall average rank (1.79). In this broader comparison, however, the differences are not statistically significant, which suggests that the three methods are very close on the standard benchmark.

The experimental analysis was extended to additional benchmark datasets proposed in the literature, under the same time limit as in the competition setting. On these datasets, our method generally improves the average results previously reported in the literature, with only marginal exceptions on a very small number of instances (Figure 1). In terms of best-known solutions, the approach improves the reported upper bounds for `test3` and `test4`, and matches the best-known values on most of the retained Udine instances.

More generally, the results suggest that mature timetabling benchmarks can still benefit from a careful re-examination of classical metaheuristics when supported by modern algorithm configuration tools and efficient implementations.

References

1. Abdullah, S., Turabieh, H.: On the use of multi neighbourhood structures within a Tabu-based memetic approach to university timetabling problems. Information Sciences **191**, 146–168 (2012)
2. Ceschia, S., Di Gaspero, L., Schaerf, A.: Educational timetabling: Problems, benchmarks, and state-of-the-art results. European Journal of Operational Research **308**(1), 1–18 (2023)
3. De Coster, A., Musliu, N., Schaerf, A., Schoisswohl, J., Smith-Miles, K.: Algorithm selection and instance space analysis for curriculum-based course timetabling. Journal of Scheduling **25**, 35–58 (2022)
4. Di Gaspero, L., Schaerf, A.: Multi-neighbourhood local search with application to course timetabling. In: Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT-2002), selected papers. Lecture Notes in Computer Science, vol. 2740, pp. 262–275. Springer (2003)
5. Kiefer, A., Hartl, R.F., Schnell, A.: Adaptive large neighborhood search for the curriculum-based course timetabling problem. Annals of Operations Research **252**(2), 255–282 (2017)
6. McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A.J., Di Gaspero, L., Qu, R., Burke, E.K.: Setting the research agenda in automated timetabling: The second international timetabling competition. INFORMS Journal on Computing **22**(1), 120–130 (2010), https://www.eeecs.qub.ac.uk/itc2007/
7. Müller, T.: ITC2007 solver description: a hybrid approach. Annals of Operations Research **172**(1), 429–446 (2009)

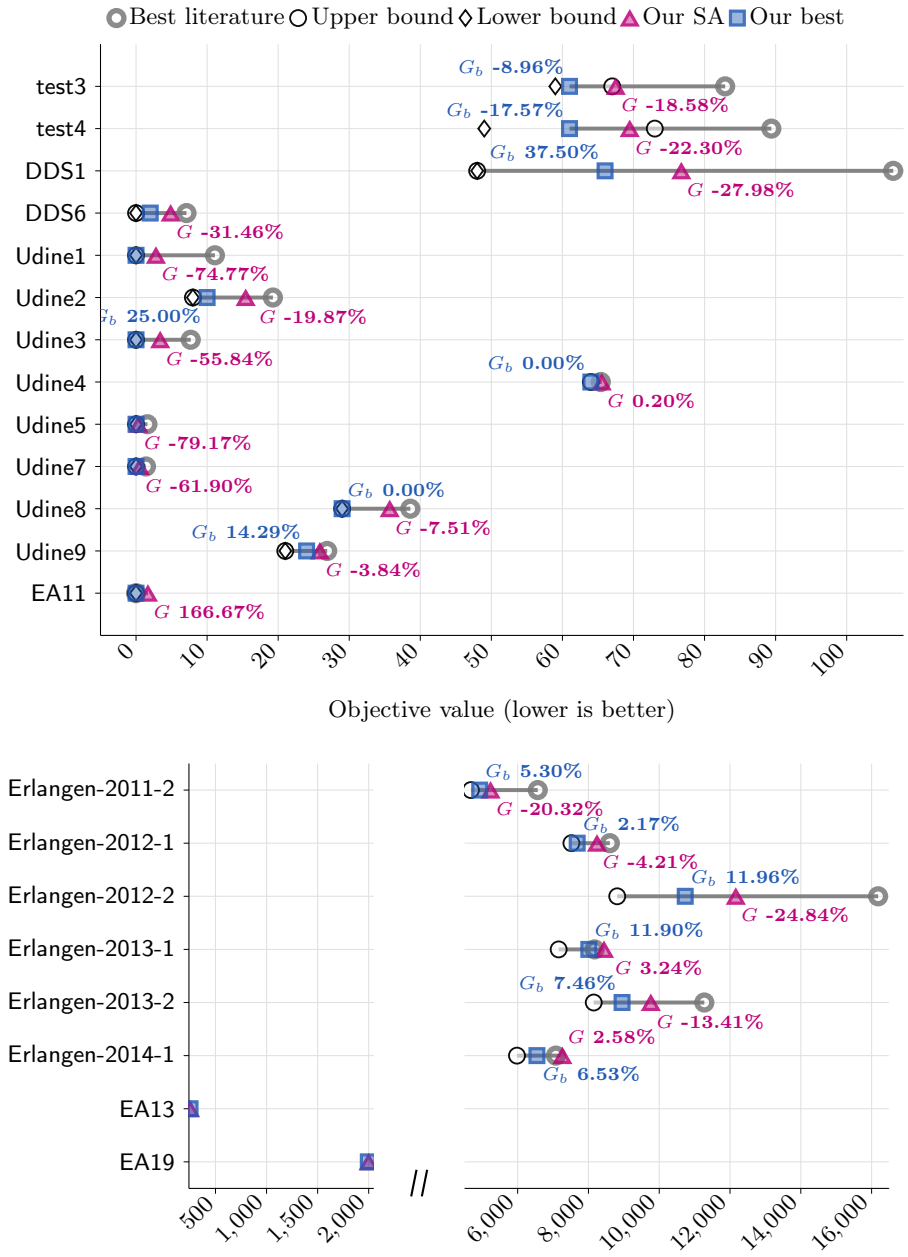


Fig. 1: Comparison plots for the test, DDS, Udine, EasyAcademy (EA), and Erlangen datasets. Symbols denote literature/known best values, lower bounds, and the average and best values of our SA; G and G_b report the corresponding gaps when available.

A Hyperbolic Multi-Strategy Search Framework for Flexible Job Shop Scheduling

Wirawati Dewi Ahmad, Azuraliza Abu Bakar, and Mohd Nor Akmal Khalid

Center for Artificial Intelligence Technology,
Faculty of Information Science and Technology,
Universiti Kebangsaan Malaysia, Bangi, Malaysia

Abstract. The Flexible Job Shop Scheduling Problem (FJSP) requires simultaneous machine assignment and operation sequencing decisions, resulting in a large combinatorial search space. Existing swarm-based scheduling approaches often rely on homogeneous initialization and fixed search behaviours, which may limit search diversity and reduce search effectiveness. This paper proposes a hyperbolic multi-strategy search framework for single-objective FJSP. The framework employs a nonlinear hyperbolic representation to improve population diversity and search-space coverage, together with adaptive operator switching among cosine exploitation, hyperbolic intensification, and Lévy-flight exploration. Candidate solutions are transformed into feasible schedules using random-key encoding and earliest finishing time (EFT) decoding. Preliminary ablation and comparative studies on representative Brandimarte benchmark instances demonstrate competitive makespan performance across different scheduling scales.

Keywords: Flexible Job Shop Scheduling · Scheduling Optimization · Adaptive Search Dynamics · Swarm Optimization

1 Introduction

The Flexible Job Shop Scheduling Problem (FJSP) is a widely studied production scheduling problem due to its practical importance in modern manufacturing systems. The simultaneous determination of machine assignments and operation sequences creates a large combinatorial search space with strong interactions between routing and sequencing decisions, resulting in an irregular search landscape and numerous local optima [3, 1]. Consequently, effective diversification and adaptive search regulation are essential for generating high-quality schedules.

Swarm-based metaheuristics have demonstrated competitive performance for FJSP due to their ability to explore complex scheduling spaces. However, generating high-quality schedules for medium- and large-scale instances remains challenging, motivating the development of more effective search mechanisms. Search diversity and the balance between exploration and exploitation play a critical role in the performance of population-based optimization algorithms. Homogeneous population initialization may limit the diversity of candidate schedules,

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

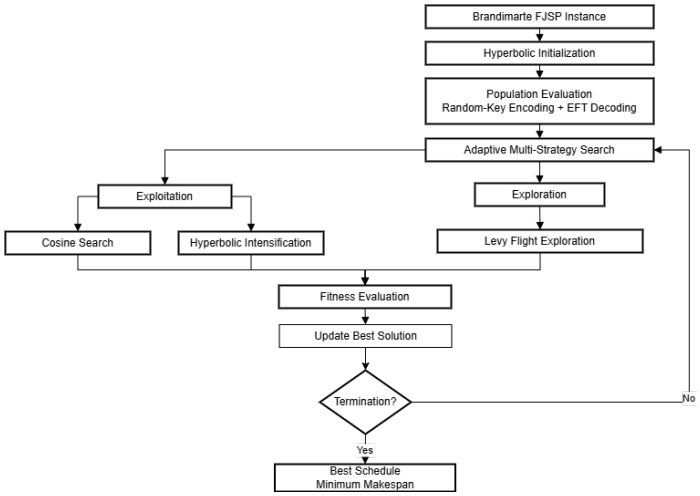


Fig. 1. Overview of the proposed hyperbolic multi-strategy search framework.

while fixed search behaviours may reduce the ability to adapt search intensity throughout the optimization process. These limitations motivate the investigation of nonlinear search representations and adaptive search strategies for FJSP optimization. To address these challenges, this study proposes a hyperbolic multi-strategy search framework that combines nonlinear hyperbolic representation, adaptive search regulation, and EFT-based schedule construction. The main contributions are: (1) a nonlinear hyperbolic representation to improve population diversity and search-space coverage; (2) an adaptive multi-strategy search framework incorporating cosine exploitation, hyperbolic intensification, and Lévy-flight exploration; and (3) a random-key and EFT-based schedule construction mechanism for generating feasible FJSP solutions.

In this study, the objective is to minimize the makespan while satisfying operation precedence, machine capacity, and machine assignment constraints. The objective function is defined as:

$$C_{\max} = \max_{i \in J} C_i \tag{1}$$

where C_i denotes the completion time of job i.

2 Hyperbolic Multi-Strategy Search Framework

The proposed framework addresses the FJSP by integrating heterogeneous initialization, adaptive search regulation, and schedule construction mechanisms.

Figure 1 presents the proposed framework. Hyperbolic initialization (Eq. 2) is used to improve population diversity and search-space coverage. During optimization, the adaptive parameter in Eq. 3 regulates cosine exploitation, hy-

Table 1. Ablation study results on representative Brandimarte benchmark instances.

Instance	AH-SCSA		AH-SCSA$_2$		AH-SCSA$_1$	
	Best	Avg	Best	Avg	Best	Avg
Mk01	39	40.0	39	41.7	39	41.9
Mk05	174	174.0	187	191.7	186	191.1
Mk08	523	523.0	531	548.0	534	548.4

perbolic intensification, and Lévy-flight exploration. Candidate solutions are decoded using random-key encoding and EFT decoding, and evaluated using the makespan objective in Eq. 1.

For exploitation, candidate solutions are refined using the original cosine-based search mechanism inherited from SCSA together with the proposed hyperbolic intensification operator(Eq. (5)). Lévy-flight perturbation is employed to preserve diversification and reduce stagnation Eq. (6).

$$X_{i,j} = lb_j + \frac{\tanh(\alpha \cdot rand)}{\tanh(\alpha)} (ub_j - lb_j) \tag{2}$$

where α controls the dispersion intensity, $rand \in [0, 1]$, and lb_j and ub_j denote the lower and upper search bounds, respectively.

$$R(t) = SM \left(1 - \frac{2t}{T + t}\right) \tag{3}$$

$$X_i^{t+1} = X_{\text{best}} - R(t)\, r \left(X_{\text{best}} - X_i^t\right) \cos(\theta) \tag{4}$$

$$X_i^{t+1} = X_i^t + \tanh(R(t))\; r_1 \left(2r_2 X_{\text{best}} - X_i^t\right) \tag{5}$$

$$X_i^{t+1} = X_i^t + \lambda, \text{Levy}(\beta) \tag{6}$$

where λ controls the step size and Levy(β) denotes a Lévy-distributed random walk.

3 Preliminary Experimental Results

The proposed framework was evaluated on three representative Brandimarte instances: Mk01, Mk05, and Mk08, representing small-, medium-, and large-scale scheduling problems. These instances enable the evaluation of solution quality and scalability across different scheduling complexities.

Table 1 compares three variants of the proposed framework. *AH-SCSA*$_1$ represents the baseline SCSA with only the discretization mechanism, while *AH-SCSA*$_2$ additionally incorporates heterogeneous initialization. The complete *AH-SCSA* further includes the proposed adaptive search operators. This comparison evaluates the contribution of population diversity and adaptive search mechanisms to FJSP scheduling performance.

Table 2. Comparative study results on representative Brandimarte benchmark instances.

Instance	AH-SCSA		SS-GWO[6]		DBO+SA [5]		AOOA [7]		ISCSO [2]	
	Best	Avg	Best	Avg	Best	Avg	Best	Avg	Best	Avg
Mk01	**39**	40.0	**39**	–	40	41.3	45	–	42	42.6
Mk05	**174**	**174.0**	179	–	175	177.6	197	–	179	184.3
Mk08	**523**	523.0	541	–	**523**	526.4	**523**	–	**523**	523.1

Additional comparisons with representative swarm-based and recent FJSP optimization methods are included. The preliminary results shown in table 2 indicate that the proposed framework achieves competitive makespan values across different problem scales. These results indicate that the proposed hyperbolic multi-strategy search framework is capable of producing high-quality schedules across different problem scales.

4 Conclusion

This paper proposed a hyperbolic multi-strategy search framework for the single-objective Flexible Job Shop Scheduling Problem. The framework combines hyperbolic initialization, cosine exploitation, hyperbolic intensification, Lévy-flight exploration, and EFT-based schedule construction. Preliminary results on selected Brandimarte benchmark instances demonstrate competitive scheduling performance and highlight the potential of combining adaptive search regulation with nonlinear search representations for scheduling optimization.

References

1. Dauzère-Péres, S., Ding, J., Shen, L., Tamssaouet, K.: The flexible job shop scheduling problem: A review. European Journal of Operational Research (2024)
2. Li, D., Hou, J., Zhang, Y., Fu, J.: An improved sand cat swarm optimization algorithm for flexible job shop scheduling problem. Proceedings of SPIE 13226 (2024)
3. Momenikorbekandi, A., Kalganova, T.: Intelligent scheduling methods for optimisation of job shop scheduling problems. Electronics 14 (2025)
4. Qawqzeh, Y., Alharbi, M.T., Jaradat, A., Sattar, K.N.A.: A review of swarm intelligence algorithms deployment for scheduling optimization. PeerJ Computer Science 7 (2021)
5. Yao, S., Guo, Y., Yang, B., Lv, Y., Ceccarelli, M., Dai, X., Carbone, G.: Single-objective flexible job-shop scheduling problem based on improved dung beetle optimization. STEM Education 4, 299–327 (2024)
6. Zhou, K., Tan, C., Zhao, Y., Yu, J., Zhang, Z., Wu, Y.: Research on solving flexible job shop scheduling problem based on improved GWO algorithm. Neural Processing Letters 56 (2024)
7. M. A. Al-Sharqi, A. T. Sadiq, and S. O. Al-Mamory. Flexible Job Shop Scheduling Problem-Solving Using Apiary Organizational-Based Optimization Algorithm. *ARO–The Scientific Journal of Koya University*, 12(2):94–106, 2024.

”

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Explaining Infeasibility in the Test Laboratory Scheduling Problem

Aida Aliu, Florian Mischek, and Nysret Musliu

DBAI, TU Wien, Vienna, Austria
{aida.aliu, florian.mischek, nysret.musliu

Keywords: Infeasibility Explanations, Test Laboratory Scheduling Problem, Counterfactual Explanations, Constraint Programming

1 Introduction

Scheduling problems typically consist of numerous constraints that are often conflicting. This may prevent scheduling systems from generating feasible solutions, resulting in scheduling infeasibility. Scheduling infeasibility in automated scheduling systems refers to a situation in which the system cannot generate a valid schedule that satisfies all specified constraints and requirements. Several factors can contribute to scheduling infeasibility, including resource constraints, temporal constraints, complexity of constraints, and incomplete or incorrect input data

The lack of explanations for failures to generate feasible solutions undermines the primary purpose of automated scheduling systems designed to support human users. In such cases, users are left with no solution and have to manually check for reasons behind the infeasibility, which can be very time inefficient. Therefore, a vital challenge of these systems is generating reasons that explain why the system cannot find a feasible solution under certain specifications.

We investigate generating infeasibility explanations for a specific scheduling problem, namely the Test Laboratory Scheduling Problem (TLSP), introduced by [6]. The TLSP is an NP-complete scheduling problem, a real-world scheduling problem inspired by the operational needs of industrial test laboratories. It extends the classical Resource-Constrained Project Scheduling Problem (RCPSP) [2] by introducing additional complexity through task grouping, diverse resource types, and real-world constraints like employee qualifications and equipment availability. Solvers for the TLSP have already been successfully deployed in real-world industrial labs. However, in practice, users encountered challenges due to the lack of explainability when schedules turned out to be infeasible, making it difficult to understand and resolve conflicts.

By adapting existing approaches from the literature [5, 4, 3], we developed an infeasibility explainer for the TLSP, which to the best of our knowledge, is the first explainer specifically designed for this problem. Our approach integrates two distinct explainers, each supporting different types of explanations. These explainers are designed to generate human-readable outputs that effectively identify the constraints causing infeasibility while also providing valuable

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

insights for restoring feasibility. The first explainer is based on conflict detection, utilizing minimal unsatisfiable sets (MUS) of constraints and maximal satisfiable sets (MSS) to describe the source of infeasibility. Restoring feasibility is achieved through minimal correction sets (MCS). The second explainer supports counterfactual explanations, suggesting minimal changes required to restore feasibility, and, building on an existing approach, introduces novel weighting and blocking techniques. The explanations were collaboratively constructed with a real-world TLSP domain experts, incorporating their insights on common sources of conflicts.

Our approach was implemented using a tailored TLSP constraint programming model in OR-Tools. Additionally, we developed a configurable interface that improves usability by allowing users to configure the explainer's settings and run multiple configurations. This practical solution simplifies the diagnosis of infeasibility and provides suggestions for restoring feasibility, significantly reducing the need for time-consuming trial-and-error methods.

We evaluated the explainer on real-world settings, assessing its efficacy in identifying common sources of infeasibility and evaluating its utility in practical settings. The experiments conducted included both qualitative analysis and efficiency assessments. We demonstrated various use cases, presenting the types of explanations generated and evaluating their effectiveness and computational performance. We refer the reader to [1] for extensive experimental evaluation and details on the explanations.

The infeasibility explainer has been successfully deployed in an industrial setting, where it effectively aids in resolving real-world scheduling infeasibilities.

References

1. Aliu, A.: Investigating explanations of infeasibility for test laboratory scheduling problems. Master Thesis, TU Wien (2025), https://doi.org/10.34726/hss.2025.115143
2. Brucker, P., Drexl, A., Möhring, R.H., Neumann, K., Pesch, E.: Resource-constrained project scheduling: Notation, classification, models, and methods. Eur. J. Oper. Res. **112**(1), 3–41 (1999), https://doi.org/10.1016/S0377-2217(98)00204-5
3. Gupta, S.D., Genc, B., O'Sullivan, B.: Finding counterfactual explanations through constraint relaxations. CoRR **abs/2204.03429** (2022), https://doi.org/10.48550/arXiv.2204.03429
4. Leo, K., Tack, G.: Debugging unsatisfiable constraint models. In: CPAIOR 2017. LNCS, vol. 10335, pp. 77–93. Springer (2017), https://doi.org/10.1007/978-3-319-59776-8_7
5. Liffiton, M.H., Sakallah, K.A.: Algorithms for computing minimal unsatisfiable subsets of constraints. J. Autom. Reason. **40**(1), 1–33 (2008), https://doi.org/10.1007/s10817-007-9084-z
6. Mischek, F., Musliu, N.: A local search framework for industrial test laboratory scheduling. Ann. Oper. Res. **302**(2), 533–562 (2021), https://doi.org/10.1007/s10479-021-04007-1

”

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Fairness-aware opponent assignment in sports timetabling with fixed home dates

Ali Hassanzadeh[1][0000−0001−8475−8676], Dries Goossens[2][0000−0003−0224−3412], and John Turner[3][0000−0002−5835−1247]

[1] Centre for AI and Decision Sciences, Alliance Manchester Business School, University of Manchester, UK. ali.h@manchester.ac.uk
[2] Ghent University, Department of Business Administration and Operations Management, Belgium. dries.goossens@ugent.be
[3] Paul Merage School of Business, University of California, USA. john.turner@uci.edu

1 Introduction

Sports timetables are often judged by whether they satisfy logistical and commercial requirements, but they are also judged by whether they are fair. A common complaint from teams, fans, and the media is that some clubs face unusually difficult stretches of opponents within a short period, while others experience a smoother sequence of fixtures. This issue is especially important in leagues where short runs of difficult matches can strongly affect performance, momentum, and league position.

In this extended abstract, we study a fairness-aware timetabling problem in which the dates of home games are treated as fixed, but the assignment of opponents to those dates is to be optimized as the season progresses. This setting is motivated by the fact that, in many real leagues, home dates need to be fixed long beforehand because of venue availability, city operations, broadcast planning, and other logistical constraints. We therefore focus on a more practical question: if the home calendar is fixed, how much can fairness still be improved by dynamically choosing the opponent assignment?

2 Related Literature

Optimizing sports schedules with the requirement that a team should not play against extremely weak or extremely strong teams in consecutive rounds has been linked with fairness and studied from a theoretical perspective (see [2], [3], and [7]). In an incomplete round robin tournament (where teams face different sets of opponents), a team's strength of schedule is calculated based on the set of opponents each team faces, and their performance (see e.g. [5] for a discussion of metrics). However, even in (complete) round robin tournaments, where each team faces each opponent the same number of times, the order in which the opponents are faced has been shown to be significant (see e.g. [4]). Dynamic scheduling is rather uncommon in sports, but has been discussed for e-sports ([6]) and the NFL ([1]).

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

3 Our Contributions

Our framework measures fairness through the idea of tough stretches. Instead of looking only at average strength of schedule across a whole season, we examine rolling windows of time or games and ask whether some teams are exposed to clusters of especially strong opponents. In a deterministic formulation, each team's strength is assumed to be known at the time of scheduling (typically estimated from historical data, preseason projections, or current performance indicators). Given the team strength measures, as a first step, we optimize the schedule to reduce the worst stretch of strong opponents across the league. This gives a natural min-max fairness objective and provides a useful benchmark for understanding how much imbalance is present in an existing schedule.

In reality, team strengths are stochastic and manifest themselves as the results unfold. Rather than minimizing an *expected unfairness* measure, we propose that the league fixes the home dates for the entire season (ensuring logistical stability for venues, cities, and broadcasters), but announces the opponent assignments in smaller batches (e.g., 5–10 games at a time), instead of releasing the entire timetable at once. At each batch release, the most recent estimates of team strength are treated deterministically, the fairness model is re-solved for the remaining games, and the new batch of matchups is published. This effectively forms a stochastic program with recourse, allowing the schedule to adapt as uncertainty unfolds, but without disrupting existing logistical commitments.

We apply this idea to two different settings. The first is double round-robin scheduling, motivated by association football leagues such as the English Premier League, where each team plays every other team once at home and once away. The second is asymmetrical round-robin scheduling, motivated by professional basketball, where the number and pattern of meetings between pairs of teams are not fully symmetric. In both settings, the set of all matches to be played is fixed; what changes is the timing of which opponent is assigned to each fixed home slot. Studying these two tournament formats helps us understand in which type of competition dynamic fairness-aware scheduling has the greatest value.

Our early computational results suggest that meaningful fairness improvements are possible even under strong practical restrictions. At the same time, the work highlights an important trade-off between flexibility and computation. Allowing more freedom in reassignment generally improves the quality of the best timetable found, but it also increases model difficulty. This makes the problem interesting both from a practical timetabling point of view and from an optimization perspective.

At PATAT, we aim to present the modeling framework, discuss alternative fairness definitions, and share initial computational evidence from basketball and football leagues. We hope this opens a broader discussion on fairness-aware timetabling in sports, especially in settings where the full schedule cannot be rebuilt freely but targeted improvements may still be realistic and valuable.

4 Mathematical Model

We present a simple deterministic model for a double round-robin league. Let T be the set of teams. For each team $i \in T$, let H_i be the set of fixed home slots of team i, and let r_{is} denote the round in which home slot $s \in H_i$ is played. Let q_j be the strength value of team $j \in T$, obtained for example from historical data or current performance indicators. We define a binary decision variable x_{isj} which is one if team i hosts team j in home slot s, and zero otherwise. The basic assignment constraints are:

$$\sum_{j \in T \setminus \{i\}} x_{isj} = 1 \qquad \forall i \in T,\ \forall s \in H_i, \tag{1}$$

$$\sum_{s \in H_i} x_{isj} = 1 \qquad \forall i, j \in T,\ i \neq j. \tag{2}$$

Constraint (1) ensures that each fixed home slot receives exactly one opponent. Constraint (2) ensures that each team i hosts each other team j exactly once over the season. To measure fairness, we consider rolling windows of L rounds. For each team i and each starting round k, let W_{ik} be the set of home slots of team i whose rounds belong to $\{k, k+1, \ldots, k+L-1\}$. The toughness of this window is

$$\sum_{s \in W_{ik}} \sum_{j \in T \setminus \{i\}} q_j x_{isj}.$$

We introduce a variable z representing the worst window toughness across the league, and minimize it:

$$\min \quad z \tag{3}$$

$$\text{s.t.} \quad \sum_{s \in W_{ik}} \sum_{j \in T \setminus \{i\}} q_j x_{isj} \leq z \qquad \forall i \in T,\ \forall k, \tag{4}$$

$$x_{isj} \in \{0, 1\} \qquad \forall i \in T,\ \forall s \in H_i,\ \forall j \in T \setminus \{i\}. \tag{5}$$

The objective (3)–(5) minimizes the maximum strength accumulated by any team over any rolling window of length L. This provides a simple min-max notion of fairness based on avoiding clusters of strong opponents. This formulation is intentionally kept simple. It is meant to illustrate the main idea of fairness-aware opponent assignment when home dates are fixed, rather than to capture every practical feature of a real sports timetable. In particular, the model uses fixed team strength values, focuses only on opponent assignment, and measures fairness through a basic rolling-window criterion.

A few additional points are worth noting. First, in a fuller version of the problem, fairness should ideally be evaluated over the full sequence of matches for each team, including both home and away games, even though the decision variables act only on the assignment of opponents to fixed home slots. Second, in a deterministic setting, the strength parameters are treated as fixed inputs at the time of optimization; dynamic updates of team strength belong more naturally

to a rolling-horizon or stochastic extension. Third, one may also want to limit how far a match can move away from its original date or opponent pattern, in order to preserve practical similarity to the published schedule.

We do not include these extra details here because of the short format of the conference extended abstract. They are not ignored, but deferred. A more complete treatment, including full home-and-away sequence effects, richer fairness definitions, and dynamic or stochastic extensions, will be discussed in the presentation and developed further in the full paper and the broader research project.

5 Preliminary Results

To illustrate the potential of the model, we applied it to three English Premier League (EPL) seasons (2021–2022, 2022–2023, and 2023–2024) using a 14-day rolling window, which can be viewed as roughly corresponding to two or three rounds of matches. Across all nine tested instances (three seasons and three strength definitions), the optimized schedule, as shown in Figure 1, improved the worst 14-day stretch relative to the original schedule. The reduction ranges from 27.9% to 42.4%, with an average improvement of 34.6%. This suggests that even under the strong restriction that home dates remain fixed, there is still meaningful room to reduce concentrated runs of strong opponents.

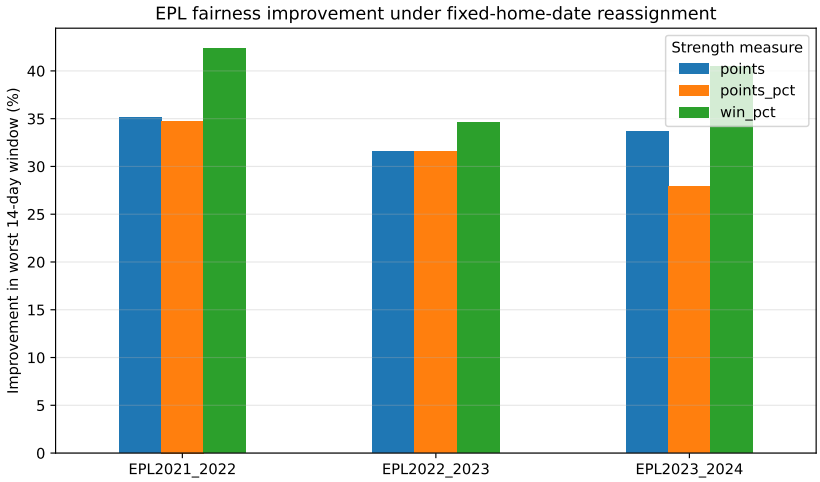


Fig. 1. Percentage improvement in the worst 14-day window for three EPL seasons under three team-strength definitions. In every tested case, the optimized assignment improves upon the original schedule.

References

1. Bouzarth, E.L., Cromer, A.W., Fravel, W.J., Grannan, B.C., Hutson, K.R.: Dynamically scheduling NFL games to reduce strength of schedule variability. Journal of Sports Analytics **6**(4), 281–293 (2020)
2. Briskorn, D.: Combinatorial properties of strength groups in round robin tournaments. European Journal of Operational Research **192**(3), 744–754 (2009)
3. Briskorn, D., Knust, S.: Constructing fair sports league schedules with regard to strength groups. Discrete Applied Mathematics **158**(2), 123–135 (2010)
4. Csató, L., Molontay, R., Pintér, J.: Tournament schedules and incentives in a double round-robin tournament with four teams. International Transactions in Operational Research **31**(3), 1486–1514 (2024)
5. DeHollander, A., Karwan, M.: Improving strength of schedule metrics in sports scheduling. Journal of Quantitative Analysis in Sports **22**(1), 53–78 (2026)
6. Dong, Z.L., Ribeiro, C.C., Xu, F., Zamora, A., Ma, Y., Jing, K.: Dynamic scheduling of e-sports tournaments. Transportation Research Part E: Logistics and Transportation Review **169**, 102988 (2023)
7. Zeng, L., Mizuno, S.: Constructing fair single round robin tournaments regarding strength groups with a minimum number of breaks. Operations Research Letters **41**, 506–510 (2013)

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Fair allocation of over- and understaffed shifts in nurse rostering

Lisa Garcia Tercero[1,2][0009-0005-9058-0508], Pieter Smet[1][0000-0002-3955-7725], Dries Goossens[2][0000-0003-0224-3412], and Greet Vanden Berghe[1][0000-0002-0275-5568]

[1] KU Leuven, Department of Computer Science, CODeS, Gebroeders de Smetstraat 1, 9000 Ghent, Belgium
{Lisa.GarciaTercero,Pieter.Smet}@kuleuven.be,
Greet.VandenBerghe@cs.kuleuven.be
[2] Ghent University, Faculty of Economics and Business Administration, Tweekerkenstraat 2, 9000 Ghent, Belgium
Dries.Goossens@ugent.be

1 Introduction

The increasing problem of nurse shortages worldwide poses a high threat to the quality of healthcare [1, 4, 5]. To prevent nurses from leaving the healthcare sector, it is crucial that their job satisfaction is taken into account during the rostering process. An important factor in this matter is to include fairness considerations. Fuchs, Schimmelpfeng and Brunner [3] provide an extensive review of the literature regarding fairness in hospital personnel scheduling, including a wide range of fairness measures to take into account such as the number of night shifts worked, the number of weekends worked and the total number of hours worked. While all of these measures are important to consider, some of them are already implicitly managed by laws and contracts. For example, the number of hours worked (on average) per week is usually determined in an employee's contract, and any overtime hours should be compensated. However, nurse shortages can also lead to understaffed shifts, increasing the workload for nurses on those shifts without any financial compensation. Besides job dissatisfaction, this can lead to reduced focus and quality of care on those shifts and an increased burnout risk for nurses in the long term. Despite these implications, fairness with regard to shift coverage has not been considered before in the literature, possibly because the impact of shift coverage violations is usually considered to be purely financial.

It is not obvious how to assess fairness with regard to understaffed shifts. As in many problems, improving fairness comes with a possible decrease in efficiency. An efficient roster avoids understaffed shifts altogether, but also needs to take into account that more shortages within a single shift create disproportionally more workload for the remaining nurses compared to distributing the same number of shortages over multiple shifts. In addition, it is not efficient to assign

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

more nurses to a shift than necessary, although it may be required in practice to fulfill all contractual obligations. In this extended abstract we discuss how to model efficiency and fairness for nurse rostering in terms of shift coverage, and we consider the trade-off between both.

2 Problem description and methodology

Table 1 denotes the key sets and parameters of a nurse rostering instance. Note that these can be extended to include any kind of hard constraints, but in this extended abstract we focus on the objectives related to shift coverage. An example of a formulation including additional constraints can be found in [2].

Table 1. Sets and parameters for a nurse rostering problem.

Sets	
D	Days in the decision horizon (index d)
S	Shift types (index s)
N	Nurses (index n)
Parameters	
δ_{ds}	Preferred number of nurses assigned to shift type s on day d
ω_{ds}^{over}	Cost per nurse exceeding the demand δ_{ds} for shift type s on day d
ω_{ds}^{under}	Cost per nurse falling short of the demand δ_{ds} for shift type s on day d
Decision variables	
x_{nds}	1 if nurse n is assigned to shift type s on day d, 0 otherwise
u_{dsc}	1 if there is a shortage of exactly c nurses on shift type s on day d, 0 otherwise ($0 < c \leq \delta_{ds}$)
v_{ndsc}	1 if there is a shortage of exactly c nurses on shift type s on day d and nurse n is assigned to that shift, 0 otherwise ($0 < c \leq \delta_{ds}$)
o_{ds}	number of nurses exceeding the demand δ_{ds} for shift type s on day d

Below we formulate the linking constraints between the decision variables. The first set of constraints links the overstaffing variables o and the understaffing variables u to the assignment variables x.

$$\sum_{n \in N} x_{nds} - o_{ds} + \sum_{c=1}^{\delta_{ds}} c \cdot u_{dsc} = \delta_{ds} \qquad \forall d \in D, s \in S \quad (1)$$

The other constraints link v to x and u, defining $v_{ndsc} = u_{dsc} \cdot x_{nds}$.

$$v_{ndsc} \leq u_{dsc} \qquad \forall n \in N, d \in D, s \in S, c \in [1, \ldots, \delta_{ds}] \quad (2)$$

$$v_{ndsc} \leq x_{nds} \qquad \forall n \in N, d \in D, s \in S, c \in [1, \ldots, \delta_{ds}] \quad (3)$$

$$v_{ndsc} \geq u_{dsc} + x_{nds} - 1 \qquad \forall n \in N, d \in D, s \in S, c \in [1, \ldots, \delta_{ds}] \quad (4)$$

The objective function includes both an efficiency and a fairness objective. The efficiency objective aims to minimize the total staffing shortage over all shifts, taking into account that the greater the shortage on a given shift, the heavier the workload becomes for the remaining nurses covering that shift. This should be reflected in the objective function, which is why the penalty increases exponentially as more nurses fall short on a given shift. Moreover, an efficient solution also avoids overstaffed shifts. The priority balance between over- and understaffed shifts is determined by their respective weights, where ω_{ds}^{over} is usually much smaller than ω_{ds}^{under}.

$$\min \sum_{d \in D} \sum_{s \in S} \omega_{ds}^{over} o_{ds} + \sum_{d \in D} \sum_{s \in S} \omega_{ds}^{under} \left(\sum_{c=1}^{\delta_{ds}} u_{dsc} \cdot 3^c \right) \tag{5}$$

The fairness metric considers the number of understaffed shifts each nurse has to cover. From a nurse's point of view, covering an overstaffed shift does not bring any major disadvantages like covering an understaffed shift does. Therefore overstaffing is included in the efficiency objective but not in the fairness objective. The formulation of a fairness objective function is less straightforward and may impact the final solution, which is why we designed three different fairness objectives.

(1) The first fairness objective aims to minimize the exponential cost of understaffed shifts for the worst-off nurse (min-max).

$$\min \max_{n \in N} \sum_{d \in D} \sum_{s \in S} \omega_{ds}^{under} \sum_{c=1}^{\delta_{ds}} v_{ndsc} \cdot 3^c \tag{6}$$

(2) The second objective aims to minimize the difference between the best-off and the worst-off nurse (min-max spread).

$$\min \max_{n,n' \in N, n' < n} \left| \sum_{d \in D} \sum_{s \in S} \omega_{ds}^{under} \sum_{c=1}^{\delta_{ds}} v_{ndsc} \cdot 3^c - \sum_{d \in D} \sum_{s \in S} \omega_{ds}^{under} \sum_{c=1}^{\delta_{ds}} v_{n'dsc} \cdot 3^c \right| \tag{7}$$

(3) The third objective aims to minimize the aggregated difference between all pairs of nurses (min-total spread).

$$\min \sum_{n \in N} \sum_{n'=1}^{n-1} \left| \sum_{d \in D} \sum_{s \in S} \omega_{ds}^{under} \sum_{c=1}^{\delta_{ds}} v_{ndsc} \cdot 3^c - \sum_{d \in D} \sum_{s \in S} \omega_{ds}^{under} \sum_{c=1}^{\delta_{ds}} v_{n'dsc} \cdot 3^c \right| \tag{8}$$

Additionally, the instances we consider to report computational results include the following hard constraints:

- A lower and upper limit on the number of hours/days worked per nurse;
- Per shift type, an upper limit on the number of these shifts worked per nurse;
- An upper limit on the number of weekends worked per nurse;

- A lower and upper limit on the number of consecutive days worked;
- A lower limit on the number of consecutive days off;
- A set of forbidden shift sequences (e.g., a night shift must not be immediately followed by an early shift);
- Each nurse has a fixed set of days off, on which they must not be assigned.

The size of the model is almost completely determined by the sizes of D, S and N. While the planning horizon can range from two weeks up to a whole year, in practice planning horizon of one or three months are common. The number of shift types per day is often equal to four (early, day, late, night), and a hospital department usually consists of 10 to 30 nurses.

Solving such an IP-model including the efficiency and one of the fairness objectives with different priorities allows us to infer the Pareto front of these two objectives. This provides insight in the "cost" of efficiency when improving fairness and vice versa. While hospitals may initially want to focus on efficiency to optimize the day-to-day workflow, an unfair distribution of these understaffed shifts carries implicit costs in the longer term, such as the risk of nurses having a burnout due to excessive workload. Moreover, the shape of the Pareto front is impacted by the modeling choices, such as the fairness objective function, the formulation of exponential penalties, and the assumption of which constraints are hard and which are soft. Investigating the impact of these specific choices on the solution outcome provides further valuable insights for the planning department of hospitals.

References

1. Chan, Z.C., Tam, W.S., Lung, M.K., Wong, W.Y., Chau, C.W.: A systematic literature review of nurse shortage and the intention to leave. Journal of Nursing Management **21**(4), 605–613 (2013). https://doi.org/https://doi.org/10.1111/j.1365-2834.2012.01437.x
2. Curtois, T., Qu, R.: Computational results on new staff scheduling benchmark instances. ASAP Res. Group, School Comput. Sci., Univ. Nottingham, Nottingham, UK, Tech. Rep (2014)
3. Fuchs, G., Schimmelpfeng, K., Brunner, J.O.: A roadmap for integrating fairness in personnel planning and scheduling in hospitals. Operations Research, Data Analytics and Logistics **45**, 200479 (2025). https://doi.org/https://doi.org/10.1016/j.ordal.2025.200479
4. Goodare, P.: Literature review: Why do we continue to lose our nurses? The Australian Journal of Advanced Nursing **34**(4), 50–56 (2017)
5. World Health Organization: Nursing and midwifery (July 2025), https://www.who.int/news-room/fact-sheets/detail/nursing-and-midwifery, accessed: 2026-03-24

The Incomplete Traveling Tournament Problem

David Van Bulck[1][0000−0002−1222−4541], Karel Devriesere[1][0000−0002−6774−849X], and Dries Goossens[1][0000−0003−0224−3412]

Ghent University, Department of Business Administration and Operations Management, Belgium. david.vanbulck@ugent.be, karel.devriesere@ugent.be, dries.goossens@ugent.be

1 Introduction

In a double round robin sports timetable, each team plays every other team once at home and once away. The Traveling Tournament Problem (TTP) seeks to construct such a timetable while minimizing the total travel distance of all teams (see also [2, 3]). It is one of the best studied sports timetabling problems, with research focusing on the development of lower bounds (see, e.g., [9]), as well as heuristic (e.g., [4]), exact (e.g., [8]), and approximation algorithms (e.g., [10]). In fact, the TTP has featured at PATAT on several occasions (e.g., [5, 6]).

In an incomplete round robin timetable, all teams play the same number of games, but not necessarily against the same subset of opponents. The choice of what games to include is left to the organizer. This format is rapidly gaining popularity, with a prominent example being the group stage of the UEFA Champions League. Applications have also emerged in non-professional youth competitions (e.g., [1, 7]).

This extended abstract discusses the incomplete Traveling Tournament Problem (iTTP), which introduces the TTP into the realm of incomplete round robin timetabling where not all teams play each other an equal number of times (e.g., like in the NBA)[1]. Given an even number of teams and an even number of rounds, with the number of rounds less than the number of teams, as well as the pairwise distances between the teams' home venues, a feasible solution to the iTTP is an incomplete round robin timetable such that:

(C1) each team plays exactly one game per round,
(C2) each team plays the same number of home and away games,
(C3) no team plays more than three home or away games in a row, and
(C4) each team meets every other team *at most* once.

The objective in the iTTP is to minimize the overall distance travelled by all teams. We assume that teams that consecutively play away travel directly between the opponent's venues, and that they start and end the season at home. As will be shown in this talk, the iTTP is $\mathcal{NP}$-hard even when restricted to only two rounds (making that (C3) does not apply). We introduce several new benchmark instances which are based on those for the classic TTP and which are available at the website of RobinX[2].

[1] An extended version is available at https://arxiv.org/abs/2603.19754.
[2] See https://robinxval.ugent.be/RobinX/travelRepo.php.

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

2 Lower Bounds

A first attempt to obtain tight lower bounds for the iTTP was made by computing an analogue of the independent lower bound (ILB; see also [9]), which computes the minimal travel distance for each team independently from all the others. In the iTTP, the ILB corresponds to solving for each team an instance of the prize-collecting vehicle routing problem, in which some customers may be skipped as long as a given total prize is collected. While the ILB is known to provide tight bounds for the classical TTP, it turns out to be surprisingly weak for the iTTP. The main reason is that it does not prevent two teams from visiting each other when they are mutual nearest neighbours, which is forbidden in the iTTP by *(C4)*. At the same time, remote teams that are not among the nearest neighbours of any other team are possibly never visited (violating *(C2)* for this remote team). To address these issues, we propose several extensions of the ILB and refer to the resulting lower bound as the Dependent Lower Bound (DLB). The DLB constructs the set of roadtrips of teams, ensuring no pair of teams meets more than once, without considering the rounds in which games are scheduled.

3 Solution Strategies

Even the smallest of the proposed benchmark instances for the iTTP seem non-trivial to solve. We experimented with a straightforward Integer Programming (IP) formulation as well as a more advanced formulation that contains a variable for each possible road trip. Moreover, we propose a hill climbing heuristic that proceeds in two phases. First, it determines the teams' home-game rounds, using a simple neighbourhood structure for which we prove that it connects the search space of all possible home-game round assignments. Given such predetermined home-game rounds, we then construct a timetable by sequentially solving bipartite matching problems (for the second step, see also [7]).

4 Computational Results

Table 1 provides computational results for two classes of the proposed benchmark instances: the National League (NL) of Major League Baseball and the National Football League (NFL). Both classes stem from the original TTP: the number before the dash in the instance name indicates the number of teams in the instance, while the number after the dash indicates the number of rounds.

The first two columns show that the ILB is indeed quite weak for the iTTP: DLB significantly improves upon it on all considered problem instances. Our proposed bound DLB is also much stronger than the LP relaxation of a naive and a trip-based IP formulation (columns LP).

Regarding the best found solutions, we see that the IP formulation with trip variables performs quite well on the NL instances. At the same time, the number of trip variables explodes for the NFL problem instances, causing this model to

Table 1. Computational experiments. IP formulations are run with 48 hours of computation time, while the matching heuristic runs on average for about 10 minutes. A '-' means the IP model could either not be fit into 64GB of memory, or the solver ran out of memory before finding a feasible solution.

			Naive IP		Trip IP		Matching Heuristic	
Instance	ILB	DLB	LP	UB	LP	UB	Best UB	Avg UB
NL16-4	15273	24464	9002	25200	22696	25076	26112	28517
NL16-8	36130	57263	13598	70448	52905	62125	66763	73147
NL16-12	55568	92580	20760	153583	91346	110037	120310	131165
NFL32-8	47506	70158	10275	115763	64912	-	85201	91932
NFL32-16	113319	175051	19857	362382	169263	-	235752	255240
NFL32-24	191083	297305	36216	651580	-	-	443518	467803

run out of memory. In these cases, the matching heuristic performs much better than the naive formulation, yet the gap with the lower bounds remains large. This once more demonstrates that the iTTP problem instances are non-trivial to solve, even though much fewer games have to be planned per team compared to the classic TTP.

Acknowledgements David Van Bulck thanks the support of Research Foundation Flanders (FWO) [23AXE35N].

References

1. Devriesere, K., Goossens, D.: Redesigning Belgian youth field hockey competitions using an incomplete round-robin tournament. INFORMS Journal on Applied Analytics **55**, 457–468 (2025)
2. Easton, K., Nemhauser, G., Trick, M.: The traveling tournament problem description and benchmarks. In: Walsh, T. (ed.) Principles and Practice of Constraint Programming — CP 2001. pp. 580–584. Springer, Berlin, Heidelberg (2001)
3. Easton, K., Nemhauser, G., Trick, M., Easton, K., Nemhauser, G., Trick, M.: Solving the travelling tournament problem: A combined integer programming and constraint programming approach. In: Burke, E., De Causmaecker, P. (eds.) Pract. Theory Autom. Timetabling IV: 4th International Conference, PATAT 2002, Gent, Belgium, August 21-23, 2002. Selected Revised Papers. pp. 100–109. Springer, Berlin, Heidelberg (2003)
4. Frohner, N., Neumann, B., Pace, G., Raidl, G.R.: Approaching the Traveling Tournament Problem with Randomized Beam Search. Evolutionary Computation **31**, 233–257 (2023)
5. Fujiwara, N., Imahori, S., Matsui, T., Miyashiro, R.: Constructive algorithms for the constant distance traveling tournament problem. In: Burke, E.K., Rudová, H. (eds.) Practice and Theory of Automated Timetabling VI. pp. 135–146. Springer, Berlin, Heidelberg (2007)

6. Henz, M.: Playing with constraint programming and large neighborhood search for traveling tournaments. In: Burke, E.K., Trick, M. (eds.) Proc. PATAT. vol. 2004, pp. 23–32 (2004)
7. Li, M., Van Bulck, D., Goossens, D.: Beyond leagues: A single incomplete round robin tournament for multi-league sports timetabling. European Journal of Operational Research **323**, 208–223 (2025)
8. Siemann, M.R., Walter, M.: A polyhedral study for the cubic formulation of the unconstrained traveling tournament problem. Discrete Optim. **46**, 100741 (2022)
9. Urrutia, S., Ribeiro, C.C., Melo, R.A.: A new lower bound to the traveling tournament problem. In: Computational Intelligence in Scheduling, 2007. SCIS'07. IEEE Symposium on. pp. 15–18. IEEE (2007)
10. Zhao, J., Xiao, M., Xu, C.: The traveling tournament problem: Improved algorithms based on cycle packing. Theoretical Computer Science **1056**, 115519 (2025)

Refining self-scheduled rosters with limited changes and blocks

Qing Chuan Ye[0000−0002−8249−9890]

ORTEC B.V., Houtsingel 5 2719 EA Zoetermeer, The Netherlands
charlie.ye@ortec.com

Abstract. Self-rostering allows healthcare workers to propose their own work schedules, increasing autonomy and engagement, but often results in rosters that do not meet operational requirements without substantial manual intervention. We present an optimization-based method that supports planners by adjusting rosters within individual change limits. The approach incorporates shift blocks and a quantified system of change points to ensure minimal modifications. To determine the smallest distance between the initial and modified rosters, we use a minimum-cost flow formulation. A pilot with a Dutch hospital will evaluate the ability of our optimizer to produce good quality rosters more efficiently.

Keywords: Nurse scheduling problem · Self-rostering · Shift blocks

1 Introduction

Self-rostering has become a popular approach in workforce scheduling in healthcare. It allows employees to influence their own work patterns, giving them a sense of control and increasing their engagement in the planning process. It also has a positive effect on their work-life balance [1] and job satisfaction [4]. Employees can create feasible rosters with regard to labor laws and agreements, as the system indicates violations and prevents the employee from submitting a roster with violations. However, it is difficult for employees to construct a roster that satisfies all duty demands without external intervention. In current practice, a planner must manually adjust the submitted individual rosters to ensure that the overall schedule meets operational requirements. This manual correction process is time-consuming, often suboptimal, and might incorporate the planner's bias.

To help the planner address these issues, we introduce an optimization-based approach that aims to satisfy the duty demands as much as possible while ensuring that the changes to each employee's self-constructed roster do not exceed predefined acceptable limits. In addition, we take into account shift blocks in the rosters, as these are important for the recognizability of an employee's schedule.

2 Shift blocks

A shift block is defined as a sequence of consecutive shifts in which the rest between the planned shifts is not more than 32 hours. Employees prefer to plan

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

shift blocks, as it makes their roster more predictable and stable. They do not like working single shifts or highly variable sequences of shifts, especially when it comes to night shifts [3]. Having the shifts blocked together may also help alleviate circadian disruption and minimize fatigue [2]. Therefore, we try to respect shift blocks in an employee's roster and aim to make only small changes. We enforce the following properties on shift blocks:

- *Minimum/maximum block length*: number of consecutive shifts in a block needs to be at least the minimum and at most the maximum block length;
- *Maximum shift types*: number of different shift types that can be present in a block. Example of shift types based on start time: *Early*, *Late*, *Night*;
- *Mandatory block*: specifies for a given shift type a collection of weekdays that needs to be included together in any block that includes one of those days. For example, if *Night* shifts have a mandatory block of [Friday, Saturday, Sunday], then if a *Night* shift is planned on a Friday, the adjacent Saturday and Sunday also must have a *Night* shift planned;
- *Extend block with shift*: adding a new shift to an existing block is preferred over adding shifts that would create a new shift block.

Furthermore, if an employee plans a block that is smaller or larger than the minimum or maximum block length, we assume that this is a deliberate choice, and we will allow this. These blocks may be changed, but if they are still present in the modified roster, we consider it to be a feasible roster.

3 Roster changes limit

Each employee has their own limit to the changes that can be made to their submitted roster. This limit is based on how much an employee has helped the overall schedule by taking on less desired shifts. Therefore, each shift in the self-rostering phase has a certain score. The less desired a shift is, the higher the score. When an employee submits their individual roster, the scores are summed up, and the planner maps the total score to a *maximum number of change points* (MP). This mapping is inverse, meaning that high total scores will lead to lower MP, and low total scores to higher MP. If an employee has submitted an infeasible roster with respect to their contract hours, the MP will be increased to accommodate the changes needed for the roster to be feasible again.

A change in the roster can be classified into different categories, each with an associated number of *change points* (CP). For example, a planner could use the following categories based on the starting times of the shifts:

- *Small change (1 CP)*:
  - changing a shift to an adjacent shift ($\leq$ 8 hours difference), e.g., an *Early* shift was changed to a *Late* shift on the same day;
  - removing a shift from an existing block;
- *Medium change (2 CP)*:

  - changing a shift to a non-adjacent shift, up to and including the same shift a day earlier or later (≤ 24 hours difference), e.g., an *Early* shift in an employee's roster was changed to a *Night* shift on the same day or to an *Early* shift on the next day;
  - adding a shift to an existing block;
- *Large change (3 CP)*:
  - adding a shift to a non-existing block, e.g., creating a new block, or adding a shift to a newly created block.

Note that any change consisting of a shift moving more than 24 hours, e.g., changing *Early* on day d to *Late* on day $d+1$ would be considered as removing (1 *CP*) and adding a shift (2 *CP* if adjacent to an existing block, 3 *CP* if not).

Finally, we consider a modified roster feasible if the total number of change points associated with the changes made in the roster for each employee is less than or equal to the employee's maximum number of change points ($\sum CP \leq MP$), in addition to remaining feasible with regard to labor laws and agreements.

4 Determining changes

Now that we have defined how many *change points* each change costs, we need to determine what the changes are between the employee's submitted roster and the modified roster. It can be difficult to determine whether a shift should be considered as added, removed, or changed from an existing shift (and from which shift). Even more so considering blocks of shifts, as a block that has moved one day later could be viewed as either all shifts have moved one day, or the first shift has been removed, and a new shift has been added at the end. We want to identify the changes that result in the fewest total change points. Therefore, we make use of a minimum-cost flow formulation, which we solve using a successive shortest augmenting paths algorithm.

For a single employee, the minimum-cost flow is defined by the directed graph $G = (V, E)$ with nodes V and edges E, see Figure 1. We call shifts in the employee's submitted roster *initial shifts* $(i_1, i_2, \ldots, i_m)$, and shifts in the modified roster *current shifts* $(c_1, c_2, \ldots, c_n)$, where m is the total number of initial shifts and n the total number of current shifts, respectively. An employee also has a limited number of jokers that they can put on a specific shift or a day off, which indicates that these are very important to the employee and, therefore, *must* be adhered to. Jokers on shifts are incorporated as *fixed* shifts, which will be present in both the initial and current shifts. Jokers on days off are incorporated as days on which shifts cannot be planned.

The RA node mediates removals and additions. An edge $i \to RA$ indicates a removal of the initial shift i, while an edge $RA \to c$ indicates an addition of the current shift c. A directed edge $i \to c$ signifies that the current shift c is obtained from the initial shift i. The source s distributes the flow to the initial shifts, and the sink t collects the flow from the current shifts. All edges have capacity 1, except the edge $s \to RA$ with capacity $p_r = \max(0, m - n)$, and $RA \to t$ with capacity $p_a = \max(0, n - m)$. This means that when there are more *current* than

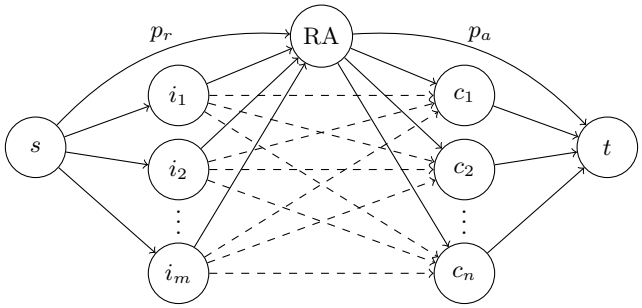


Fig. 1. Directed graph used for minimum-cost flow to determine changes between initial and current roster for a single employee.

initial shifts ($m > n$), we need to remove p_r shifts. And vice versa, when there are more *initial* than *current shifts* ($n > m$), we need to add p_a shifts.

The cost of each edge from s and to t is 0. The cost of the edges from and to the shift nodes corresponds to the change points (CP) defined in Section 3. The edges $i \to RA$ have cost 1, and $RA \to c$ have cost 2 if c is in an existing block, or cost 3 if c is in a new block. The edges $i \to c$ have cost 0 if the start time is the same, cost 1 if there is ≤ 8 hours difference in start time, or cost 2 if the difference is ≤ 24 hours. If the difference is > 24 hours, there will be no edge.

5 Customer pilot

The details of this work were established in cooperation with planners from a Dutch hospital. We are starting a pilot in 2026 using our optimizer that takes into account both shift blocks and the roster changes limit. We will compare our results to the planner's manually produced roster. Our goal is to produce a feasible roster with as few changes as possible, so that it can help the planner create a feasible roster from the employees' submitted rosters in less time.

References

1. Albertsen, K., Garde, A. H., Nabe-Nielsen, K., Hansen, Å. M., Lund, H., Hvid, H. (2014). Work-life balance among shift workers: results from an intervention study about self-rostering. *International Archives of Occupational and Environmental Health*, 87(3), 265-274.
2. Booker, L. A., Mills, J., Bish, M., Spong, J., Deacon-Crouch, M., & Skinner, T. C. (2024). Nurse rostering: understanding the current shift work scheduling processes, benefits, limitations, and potential fatigue risks. *BMC nursing*, 23(1), 295.
3. Petrovic, S., Vanden Berghe, G. (2008, August). Comparison of algorithms for nurse rostering problems. In *Proceedings of the 7th International Conference of Practice and Theory of Automated Timetabling*.
4. Webster, B., Archibald, D. (2022). Self-rostering, work-life balance and job satisfaction in UK nursing: a literature review. *Nursing Management*, 29(6).

Online Outperforms Offline: Reinforcement Learning Hyper-Heuristics for Vehicle Routing

Daniel R. Torres Ruiz[0000−0003−2640−2086], Daniel Karapetyan[0000−0003−4030−6525], Ender Ozcan[0000−0003−0276−1391], and Rong Qu[0000−0001−8318−7509]

University of Nottingham, Nottingham, UK

Keywords: Reinforcement Learning · Hyper-Heuristics · Delivery Planning

1 Introduction

Hyper-Heuristics (HHs) are a class of general-purpose methods that employ learning to reduce the cost of designing and implementing optimisation algorithms that are applicable in multiple problem domains. Selection HHs (SHHs) as a widely used type of HHs manages a set of domain-specific Low-Level Heuristics (LLHs) operators, such as mutation and hill-climbing. During the search process, SHH selects and applies these LLHs to explore the solution space.

SHHs can be divided into online and offline approaches. An online SHH learns the behaviour of LLHs and adapts the selection strategy during each run. Whereas an offline SHH follows a train-and-test paradigm: learning takes place on a training set of instances, and the learnt policy is then applied to the unseen instances. We experimentally compare these two approaches to show their strengths and weaknesses in practice. As an offline SHH, we employ Conditional Markov Chain Search (CMCS) that has been shown to produce effective search methods in several problem domains. CMCS is a single-point search method; at each iteration, it applies a selected LLH to the current solution, accepting any changes without backtracking. Choosing the next LLH depends on the previously applied LLH and its outcome, i.e., whether it improved the solution or not. The CMCS policy is learnt offline, remains the same during execution, and can potentially make the method particularly efficient under small time budgets.

Our online SHH is a new single-point search method based on Reinforcement Learning (RL). Like CMS, it also chooses the next LLH based on the last applied LLH and its outcome, accepting any change. However, its selection policy is learnt online using RL. As a result, it needs time to learn an effective policy; however, the policy adapts to the specific problem instance and the stage of the search, potentially making the method more effective under larger time budgets.

We compare these two approaches on the Vehicle Routing Problem with Time Windows (VRPTW) and show that our RL-based SHH indeed outperforms CMCS on larger time budgets, confirming the effectiveness of online learning.

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

2 Overview of methodology and related work

CMCS is a HH that selects the next LLH based on the outcome of the previously applied one through predefined transition configurations. These configurations are obtained via brute-force search, making the method sensitive to the number of LLHs as well as the size and representativeness of the training set. As a result, CMCS relies on fixed configurations whose performance may degrade when the training and test distributions differ, often requiring retuning and limiting its use in dynamic settings. CMCS has been evaluated on the Uncapacitated Facility Location [2] and Bipartite Boolean Quadratic Programming [1] problems, but no prior work evaluates it on the VRPTW in comparison with other HHs .

Selection HHs (SHHs) typically manage a set of perturbative LLHs and iteratively apply a selected LLHs to improve a solution, employing a high-level heuristic selection strategy combined with a move acceptance criterion [3]. The HyFlex framework enables the development and evaluation of general-purpose HHs by providing a standardised, problem-independent interface that encapsulates solution representation, initialisation, evaluation, and LLHs.

This general-purpose setting exposes a limitation of CMCS. HyFlex supports adaptable methods for multiple problem domains, whereas CMCS relies on offline configurations tuned for a specific domain. This creates a generality gap between general-purpose approaches and offline fixed policies tailored to particular problem characteristics. Bridging this gap requires adaptive selection mechanisms that operate online without pre-tuning and guide the decision-making process.

RL introduces such adaptivity in SHHs, but existing approaches remain limited. The RL Great Deluge HH (RLGDHH, [4]) assigns utility values to LLHs using a simple fixed reward-punishment strategy. It does not use Q-learning or a state representation; instead, it updates heuristic utilities solely based on the objective values of the solutions produced. As a result, its policy is context-independent and cannot capture interactions between heuristics during the search. This limitation motivates adopting a minimal state representation, as in CMCS, using Q-learning with TD(0) updates to guide the search process.

3 Preliminary experiments

Evaluation is conducted on all RC instances from Solomon and Homberger, with multiple seeded runs per instance and a 600-second time budget. The percentage gap metric is used and the results are compared statistically.

The configurations of the CMCS transition matrix are obtained following [1], using a broad set of instances and ten available LLHs. The best-performing configurations with 2 and 3 LLHs define the baselines CMCS-2 and CMCS-3. To assess the state representation, we include two additional variants: O_2, with only the previous LLH, and a stateless bandit-based approach.

3.1 Results

Experimental results show that online approaches consistently outperform offline fixed policies during the search. Figure 1a shows the mean gap over time for Solomon instances. All RL-based methods outperform CMCS-2 and CMCS-3. RLGDHH converges more slowly but reaches competitive performance after 500 seconds. CMCS-3 improves over CMCS-2 across the full time budget, indicating that adding a third operator improves exploration and helps avoid local minima.

Figure 1b shows similar trends on Homberger instances, although online methods require more time to surpass offline approaches due to the increased problem size. In contrast to Solomon, RLGDHH performs better than O_3 and its variants in early steps. CMCS-3 does not improve over CMCS-2, suggesting that the additional operator introduces overhead that is not compensated in larger instances. As in Figure 1c for both benchmarks, the final gap distributions reflect this CMCS-3 effect, particularly on high-demand RC Homberger instances.

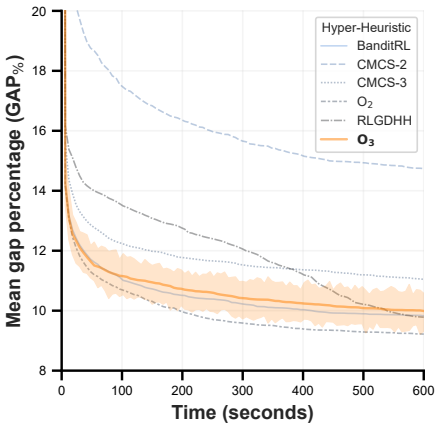
The computational cost explains this particular behaviour. In Figure 1d, online methods make fewer heuristic calls per second than offline approaches, except for the stateless variant on Solomon. The additional operator in CMCS-3 increases evaluation cost and limit the search, highlighting a trade-off between exploration and search throughput, given the instance size. Despite its simplicity, CMCS-2 remains competitive with only a mutation and a local search heuristic.

Pairwise Wilcoxon tests indicate significant differences between offline and online approaches ($p < 0.05$), but no differences among the online RL approaches ($p \geq 0.05$). Despite differing state representations, all RL methods reach similar final performance. However, their search behaviour differs from RLGDHH (Figure 1e). Crossover operators dominate across different state variants, indicating that combining the current and best solutions is the main driver of performance.

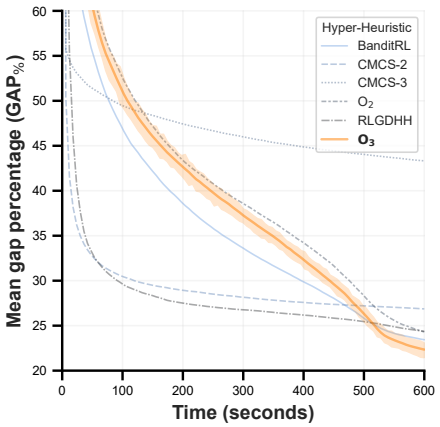
Online HHs consistently outperform offline configurations across both benchmarks, despite using fewer calls. Adaptive selection compensates for reduced search throughput by improving decision quality, highlighting the robustness of online learning and the offline policies limitations when problem scale increases.

References

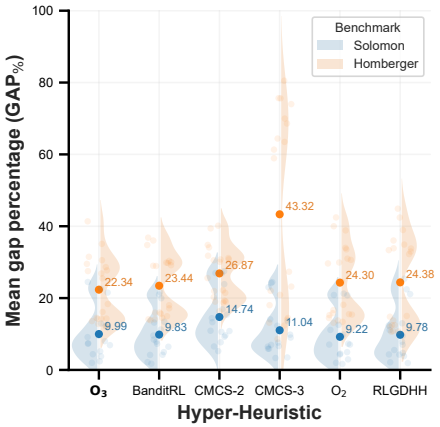
1. Karapetyan, D., Goldengorin, B.: Conditional markov chain search for the simple plant location problem improves upper bounds on twelve körkel–ghosh instances. In: Optimization Problems in Graph Theory, vol. 139, pp. 123–147. Springer International Publishing (2018). https://doi.org/10.1007/978-3-319-94830-0_7
2. Karapetyan, D., Punnen, A.P., Parkes, A.J.: Markov chain methods for the bipartite boolean quadratic programming problem. European Journal of Operational Research **260**(2), 494–506 (2017). https://doi.org/10.1016/j.ejor.2017.01.001
3. Li, C., Wei, X., Wang, J., Wang, S., Zhang, S.: A review of reinforcement learning based hyper-heuristics. PeerJ Computer Science **10**, e2141 (2024). https://doi.org/10.7717/peerj-cs.2141
4. Ozcan, E., Misir, M., Ochoa, G., Burke, E.K.: A Reinforcement Learning - Great-Deluge Hyper-Heuristic for Examination Timetabling. International Journal of Applied Metaheuristic Computing **1**(1), 39–59 (2010). https://doi.org/10.4018/jamc.2010102603



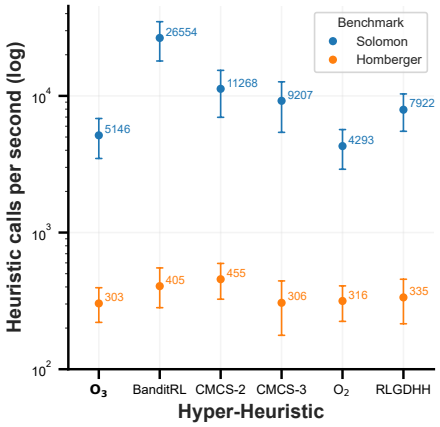
(a) Solomon (mean gap over time)



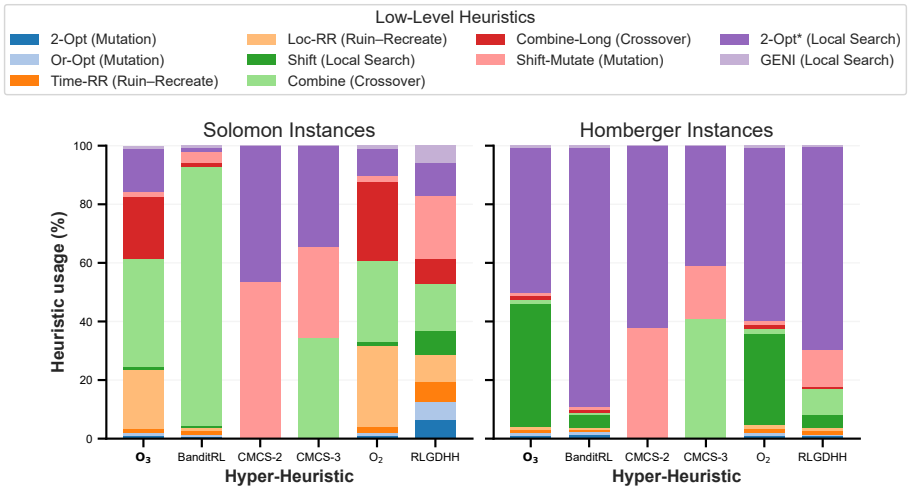
(b) Homberger (mean gap over time)



(c) Gap distributions



(d) Calls per second



(e) Usage composition

Fig. 1: Convergence profiles on (a) Solomon and (b) Homberger; (c) final percentage gap distributions; (d) heuristic calls per second; and, (e) LLH usage.

From the Field between Theory and Practice in Scheduling

Gerhard Post

Gerhard Post, PCA, Klipperweg 19, 8102 HR Raalte, The Netherlands
and
Department of Applied Mathematics, University of Twente, The Netherlands
g.f.post@utwente.nl

Keywords: Timetabling, scheduling

1 Introduction

Working as an optimization engineer is an interesting mix between scientific research and practical solutions. Of course when implementing solutions, the first intention is to use the result of scientific research. However, not all scientific research is useful in practice. For example, solving VRPs with 25 jobs by a MILP formulation and a (commercial) solver is not helpful for real life VRPs with 2000 jobs. Moreover, the model needed in practice can be more subtle than scientific research investigates. In his address at the PATAT conference of 2006, Barry McCollum discussed the gap between theory and practice, referring to the field of University Timetabling [1]. At that time, it motivated four attendees to start working on a format to describe High School Timetabling instances, encapsulating many known constraints. The instances from practice were challenging at that time and used in ITC2011 [4]; currently most of these instances can be solved to optimality. It was part of the trend that the competitions became increasingly oriented to practical problems. Thanks to this trend, the research community is studying well-documented instances, closer to practice, and with more practical value.

This contribution is based on my personal experiences of the last 25 years. My work in timetabling and scheduling began in 2001, when I started working for a big OR company. Here I worked in nurse rostering (i.e. shift scheduling), load assignment for routing (loading liquids, or solid 3D loads), multimodal transport, and task assignment. In 2019, I started working at a small company, mainly addressing service planning, a variant of Multi-Depot VRP. All the time, I kept an appointment at the university, where I researched high school timetabling, sports scheduling and personnel rostering.

My goal with this contribution is to show some challenges one encounters when trying to make an optimization suite serving a wide range of clients. Though some of these challenges look trivial from a scientific point of view, they have to be addressed to be acceptable in practice.

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

2 Areas of attention

2.1 Trust and expectations

Before a client begins using an optimization suite, usually several obstacles need to be addressed [2,3]. In large organizations, the person who decides to use the tools could be different from the persons (planners) that will be using it. As a result, there might be some resistance or reluctance from the workplace to start using it. If this hurdle is overcome, the optimization engine may produce solutions that are "good" according to the planner. In my experience this is especially true in personnel rostering, where the assigned shifts directly affect employees' daily lives. This explains why that *self rostering* approaches are frequently used in this domain.

Once the resistance is overcome, there can be planner's rules of thumb that do not always lead to optimality, and hence are violated by your algorithm that is in search for optimality. Moreover "the beam and the mote" effect can occur: seeing the violations of soft constraints in the generated solution, but overlooking these in the manual created one.

2.2 Infeasibilities

Usually an empty schedule is feasible (though not always). However, it can happen that the input schedule is not empty, but infeasible. How should this be handled? There are several ways to tackle this issue. One way (as observed in personnel rostering) is to remove shifts, until the schedule is feasible. In other areas this approach might be unfavorable, for example if the time window of the scheduled task is communicated to the customer. In this case the task will remain scheduled with the mark 'tolerated'.

2.3 Modeling

No matter how extensive your model is, there will be requests (constraints) for additional ones. These constraints can be trivial in algorithmic sense, but the data to create the constraint could be very customer specific. An example is the constraint that limits the number of tasks in a shift. But then one client requires that depending on the task type, the task is counted in a different way. It is convenient to have a 'customization' mechanism to handle such requests.

2.4 Implementation

Optimization processes can run in the background, preferably after midnight. In other situations the available runtime is preferably limited: optimization during working hours wil interfere the work of the planners, and optimization in the evening can interfere with requests that customers[1] can create online. Requests

[1] In service planning it is often the case that a customer can create a visit request.

for which the customer is waiting requests are allowed to run for only a few seconds.

For algorithms to be fast, they need specific knowledge on the state of the solution and the details of the change to execute. By contrast, generating a complete solution and passing it to a validator that rechecks all constraints from scratch may be attractive from a coding-simplicity perspective, but it is highly inefficient. Fast implementations make it possible to explore a larger part of the search space, which can significantly improve solution quality.

2.5 Support

The optimization call collects data from different (dynamic) sources that lead to the instance to be optimized. It is very helpful to have the full description of the instance saved, say, in a database. When a client asks why the optimization did or didn't do this or that, it is possible to replay the situation. Moreover, bugs (which will hopefully disappear over time) can (hopefully) be reproduced, and the corresponding instance can serve as a test case for the future. Interesting instances can be collected to serve as benchmarks for your algorithms.

References

1. B. McCollum, *A Perspective on Bridging the Gap Between Theory and Practice in University Timetabling*, PATAT 2006 Brno, Czech Republic, August 30 - September 1, 2006 Revised Selected Papers. Lecture Notes in Computer Science **3867**, Springer, 3 – 23, (2007).
2. K. McKay, W. van Wezel, and T. Waefler, *Expertise in Organizational Planning - Impact on Performance*, The Psychology of Planning in Organizations: Research and Applications, 200 – 228, (2015).
3. S. Petrovic, *"You have to get wet to learn how to swim" applied to bridging the gap between research into personnel scheduling and its implementation in practice*, Annals of Operations Research **275**, 161-179, (2019).
4. Gerhard Post, Luca Di Gaspero, Jeffrey H. Kingston, Barry McCollum, and Andrea Schaerf, *The third international timetabling competition*, Annals of Operations Research **239**, 69–75, (2016).

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Including Human Centered Objectives into Train Crew Rostering Algorithms

Veerle E. van den Hurk[1][0009-0001-7339-8334], Erwin Abbink[2], Marjan van den Akker[1][0000-0002-7114-0655], and Egon L. van den Broek[1][0000-0002-2017-0141]

[1] Department of Information and Computing Sciences, Utrecht University, Utrecht, the Netherlands
{v.e.vandenhurk,j.m.vandenakker}@uu.nl and vandenbroek@acm.org
[2] Netherlands Railways (NS), Utrecht, the Netherlands
erwin.abbink@ns.nl

Keywords: Crew Rostering · Human Centered · Preference Elicitation · Mathematical Programming

1 Introduction

Complex labor regulations and employee preferences make rostering a hard task. A large body of literature on automated approaches for rostering exists. Until recently, research mostly focused on the complex mathematical problem solving perspective, while nowadays human aspects are starting to become more prominent [9, 3, 10]. Research directions have been suggested related to this human focus: objectives that capture the needs and well-being of personnel should be included into the rostering approaches [12, 8, 10] and the effectiveness of these objectives should be evaluated [13, 5, 2].

We developed an involved Integer Linear Programming (ILP) model to generate rosters respecting the labor regulations. Next, we identified human centered objectives for rostering algorithms that are perceived by crew members as effective for reaching satisfactory rosters. These objectives were included in the ILP model and its feasibility was evaluated. We achieved this by considering a real-world case: rostering for train crew at Netherlands Railways (NS), the largest passenger railway operator in The Netherlands. To identify objectives, we looked at revealed preferences, which are observed choices made by crew members in hand made rosters. Here, we focused on objectives regarding working times by considering the rostering of *duty templates*. A duty template is characterized by a day and time window (larger than the duration of a duty) in which an employee can be scheduled to work, and acts as a placeholder for a duty in a roster [14]. Actual duties (i.e., sequences of train trips) will be assigned to a roster containing duty templates closer to the day of operation, and are guaranteed to fit within the templates. As such, template rosters are more stable when looking

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

at working times compared to duty-based rosters, since duties often are rescheduled (e.g., because of maintenance work or special events). Because of this time guarantee in template rosters, meeting working time preferences in these rosters receives more attention.

At NS, the train crew members are assigned to crew bases. Within crew bases, crew members belong to a roster group, representing their wishes or needs regarding shift types (i.e., early, late, elderly). This can change every year, and is considered fixed input in our problem setting. At each crew base, certain crew members manually create annual cyclic rosters themselves for all roster groups.

2 Methods

The ILP model developed assigns duty templates and rest days (of different types) to cyclic rosters simultaneously, contrary to other existing two-phase approaches. Furthermore, the model adheres to all Collective Labor Agreement (CLA) regulations, including norms of rest days and biweekly rest times using rolling horizons. Besides these hard constraints, no objectives are included yet. As such, the initial model corresponds to a feasibility problem. The model handles the varying start and end times of duty templates and is also able to assign reserve days in a convenient manner, such that work is guaranteed to fit in.

Little research is available relating roster characteristics to roster satisfaction [11]. If so, the characteristics evaluated do not apply to our case, as we consider them fixed input (e.g., shift length). We propose to identify objectives based on revealed preferences of the train crew. We assume that these revealed preferences can arise from comparison among hand made rosters, which are made by train crew themselves without any help of a decision support system, and rosters generated by the introduced model. To enable comparisons between the human made and model generated rosters, the model is used to generate rosters for the same instances. The two sets of rosters are compared on an initial set of roster characteristics, which we hypothesize to be (not) preferred by crew. By considering this hypothesized set resulting from literature, we mitigate the potential issue of eliciting characteristics that are not actual preferences, but rather a side-effect of something else. It is suggested that preferences of personnel regarding working times are more influenced by work-life fit and social issues compared to health aspects [1, 4]. From literature relating these issues to roster characteristics, we come up with our hypothesized set of (not) preferred characteristics and define their corresponding measures based on existing work [6].

For comparing the hypothesized characteristics between human made and model generated rosters, a multiple two-sample t-test approach does not satisfy. The characteristics are proportions, and therefore do not follow a normal distribution. Furthermore, the data has a nested structure: observations (single rosters) are nested within roster groups (i.e., a roster for a single roster group is made twice, hand made as well as model generated), and roster groups are nested within crew base. This violates the independence assumption. As a result,

the comparison between human made and model generated rosters is done using generalized linear mixed models (GLMMs) [7].

In our case, using GLMMs means we fit a linear model to our data for each roster characteristic Y. This model is allowed to have random intercepts: both crew base i and roster group j can have their own intercept to account for the nested structure in the data. Afterwards, the result is transformed using the Logit link function, to account for the distribution of the roster characteristics. This results in the following null model, in which we do not consider the effect for human made vs. model generated (condition) yet:

$$M0 : \text{Logit}(Y_{ijk}) = \alpha + \nu_i + u_{ij} + \epsilon_{ijk},$$

where Y_{ijk} is a roster characteristic for roster $k \in \{0, 1\}$ for roster group j in crew base i. The nesting of rosters (i.e., human or ILP based) within roster groups is accounted for in the model by the random intercept u_{ij}. The nesting of roster groups within crew bases is accounted for by the random intercept per crew base, ν_i. The mean value of the outcome variable over all crew bases, roster groups and rosters is denoted by α and the residual by ϵ_{ijk}. We add the roster approach to this model. Using a fixed effect, we arrive at our alternative model:

$$M1 : \text{Logit}(Y_{ijk}) = \alpha + \beta Human_{ijk} + \nu_i + u_{ij} + \epsilon_{ijk},$$

where $Human_{ijk}$ is a binary variable indicating the rostering approach, for roster k of roster group j in crew base i. Whenever the mean value of a roster characteristic significantly differs between human and ILP based rosters, we consider this a revealed preference of the train crew. This is the case whenever the second model fits our data better compared to $M0$ for a specific roster characteristic, which we test for using the likelihood ratio test and apply Bonferroni correction.

Finally, the revealed preferences are added to the ILP model as objectives using a weighted sum approach. We study the feasibility of this human centered model by running the instances again and compare the resulting rosters to the human made ones.

3 Results

Using Gurobi 13.0.0 and an Intel Core i7-1365U 13th generation (1.80 GHz) processor, the initial ILP model solved all 7 instances, including the largest crew base of 200 members, within 6 minutes.

The GLMMs revealed significant differences between the hand made and model generated rosters for the following characteristics:

- % very long working weeks (i.e., working weeks of more than 8 hours longer than contract duration),
- % > 4 consecutive work days,
- % weekend work (i.e., working both Saturday and Sunday),
- % broken weekends (i.e., working either Saturday or Sunday),

– % single days off, and
– % single work days.

All these characteristics, except for *weekend work*, occur less in the human made rosters compared to the model generated ones. Weekend work however, occurs more often in the human made rosters compared to the model generated ones, as expected when having less broken weekends.

When considering an updated ILP model including these revealed preferences, all instances but the largest are solved to optimality within 1 hour. Comparison to human made rosters shows that the model even exceeds human performance for some of the objectives. Hereby, we show that including revealed preferences leads to a model for crew rostering that both includes effective human centered goals and complies with CLA regulations, which is something that humans themselves do not always achieve.

4 Future Work

This study is a first step in the elicitation of roster preferences for train crew. Additional qualitative studies are needed to get a more complete view of the factors that humans consider when evaluating working times in rosters, as well as to evaluate to what extend human made rosters indeed reflect human preferences. Furthermore, personalized rostering, which accounts for different individuals having different preferences, is an interesting topic for further research.

Acknowledgments. The authors have received funding for this work from the Dutch Top Consortia for Knowledge and Innovation (TKI) of the Creative Industry (CLICKNL) for the project HUPLAN - the human central, under grant agreement no. TKI1906-CI21012.

References

1. Åkerstedt, T., Kecklund, G.: What work schedule characteristics constitute a problem to the individual? A representative study of Swedish shift workers. Applied Ergonomics **59**, 320–325 (2017). https://doi.org/10.1016/j.apergo.2016.09.007
2. Böðvarsdóttir, E.B., Smet, P., Vanden Berghe, G.: Behind-the-scenes weight tuning for applied nurse rostering. Operations Research for Health Care **26**, 100265 (2020). https://doi.org/10.1016/j.orhc.2020.100265
3. van den Broek, E.L.: Monitoring technology: The 21st century's pursuit of well-being? EU-OSHA Discussion paper, European Agency for Safety and Health at Work (EUOSHA), Bilbao, Spain (6 July 2017), https://osha.europa.eu/en/tools-and-publications/publications/monitoring-technology-workplace/view
4. Emmanuel, T., Griffiths, P., Lamas-Fernandez, C., Ejebu, O.Z., Dall'Ora, C.: The important factors nurses consider when choosing shift patterns: A cross-sectional study. Journal of Clinical Nursing **33**(3), 998–1011 (2024). https://doi.org/10.1111/jocn.16974

5. Fuchs, G., Schimmelpfeng, K., Brunner, J.O.: A roadmap for integrating fairness in personnel planning and scheduling in hospitals. Operations Research, Data Analytics and Logistics **45**, Article 200479 (2025). https://doi.org/10.1016/j.ordal.2025.200479
6. Härmä, M., Ropponen, A., Hakola, T., Koskinen, A., Vanttola, P., Puttonen, S., Sallinen, M., Salo, P., Oksanen, T., Pentti, J., et al.: Developing register-based measures for assessment of working time patterns for epidemiologic studies. Scandinavian Journal of Work, Environment & Health **41**(3), 268–279 (2015). https://doi.org/10.5271/sjweh.3492
7. Hox, J., Moerbeek, M., Van de Schoot, R.: Multilevel analysis: Techniques and applications. Routledge, 3 edn. (2017). https://doi.org/10.4324/9781315650982
8. Kellogg, D.L., Walczak, S.: Nurse scheduling: From academia to implementation or not? Interfaces **37**(4), 355–369 (2007). https://doi.org/10.1287/inte.1070.0291
9. Nurmi, K., Kyngäs, J.: Staff rostering: Tackling understaffing and deviation of working time from a research, business, and well-being perspective. In: Sarfraz, M., Mohsin, M. (eds.) Future Frontiers in Operations Management - Navigating the 21st Century, chap. 8. IntechOpen, London (2025). https://doi.org/10.5772/intechopen.1012314
10. Petrovic, S.: "You have to get wet to learn how to swim" applied to bridging the gap between research into personnel scheduling and its implementation in practice. Annals of Operations Research **275**, 161–179 (2019). https://doi.org/10.1007/s10479-017-2574-4
11. Piszczek, M.M., Martin, J.E., Pimputkar, A.S., Laulié, L.: What does schedule fit add to work–family research? The incremental effect of schedule fit on work–family conflict, schedule satisfaction, and turnover intentions. Journal of Occupational and Organizational Psychology **94**(4), 866–889 (2021). https://doi.org/10.1111/joop.12367
12. Uhde, A., Laschke, M., Hassenzahl, M.: Design and appropriation of computer-supported self-scheduling practices in healthcare shift work. Proceedings of the ACM on Human-Computer Interaction **5**(CSCW1), 1–26 (2021). https://doi.org/10.1145/3449219
13. Uhde, A., Schlicker, N., Wallach, D.P., Hassenzahl, M.: Fairness and decision-making in collaborative shift scheduling systems. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. p. 1–13. CHI '20, Association for Computing Machinery, New York, USA (2020). https://doi.org/10.1145/3313831.3376656
14. van Rossum, B., Dollevoet, T., Huisman, D.: Railway crew planning with fairness over time. European Journal of Operational Research **318**(1), 55–70 (2024). https://doi.org/10.1016/j.ejor.2024.04.029

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

Simultaneous Optimization of Two-Shift and Three-Shift Nurse Rosters with Infection-Risk-Based Transition

Mio Horai[1], Wei Wu[2], and Mutsunori Yagiura[1]

[1] Nagoya University, Japan
horai.mio.c7@s.mail.nagoya-u.ac.jp
[2] Shizuoka University, Japan

Abstract. Nurse shortages become particularly severe during infectious disease outbreaks, and some hospitals operating under a three-shift system are forced to make a transition to a two-shift system in emergency situations. Focusing on such situations, we propose a scheduling method that enables smooth transitions between three-shift and two-shift duty rosters based on infection risk estimates derived from simulations based on the susceptible–exposed–infectious–recovered (SEIR) model. The proposed method suppresses undesirable shift sequences and simultaneously generates both three-shift and two-shift duty rosters. This reduces handover-related infection risk while preserving roster stability. Compared with a conventional re-optimization approach, the proposed method improves the stability of duty rosters.

Keywords: Nurse rostering · Shift transition · Infection modeling

1 Introduction

Nurse shortages in Japanese healthcare settings became particularly severe during the COVID-19 pandemic. Frequent staff absences resulted in repeated revisions of duty rosters, increasing both operational and psychological burdens on nurses and schedulers.

In studies related to COVID-19, resource allocation and workforce deployment approaches based on epidemiological models such as susceptible–exposed–infectious–recovered (SEIR) model have been proposed [1]. Furthermore, in practice, work systems have been modified in response to the infection situation [2].

In Japan, many hospitals operate either a three-shift system or a two-shift system. During the COVID-19 pandemic, some hospitals changed their work systems from a three-shift system to a two-shift system in response to staff shortages. However, if such changes are not properly designed, they may lead to excessive consecutive working days and undesirable shift patterns.

In this study, we propose a unified framework that integrates infection risk prediction with mathematical optimization to simultaneously generate both three-shift and two-shift duty rosters. Transitioning to a two-shift system helps mitigate staff shortages during infectious disease outbreaks. Preparing both rosters

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

in advance enables smooth transitions and reduces infection risk by limiting shift-change contacts. It also reduces the burden of sudden schedule changes. Furthermore, the rosters generated by the proposed framework are robust even when the transition timing is uncertain due to upcoming infection conditions. For brevity, this paper focuses on transitions from a three-shift system to a two-shift system.

2 Method

In this study, we construct an infection process model based on a stochastic SEIR framework that considers healthcare workers, patients, and community exposure, and we predict infection dynamics using Monte Carlo simulation. In particular, we evaluate the probability that the infection rate among nurses first reaches a hospital-defined critical threshold (i.e., first-passage probability). We define a daily risk weight w_t for each day t based on the predicted probability, which is used to penalize undesirable shift patterns and excessive consecutive working days near likely transition periods.

Let N denote the set of nurses and T the set of days. The three-shift system consists of day, evening, and night shifts, whereas the two-shift system consists of day and night shifts (including a post-night rest period). Both systems also include shift candidates called special duties and days off. A special duty represents non-regular daytime assignments, such as staff training or temporary support in other hospital departments. Let $S^{(3)}$ and $S^{(2)}$ denote the sets of shift candidates for the three-shift and two-shift systems, respectively.

Binary variables x_{its} and y_{its} indicate whether nurse i is assigned to shift s on day t in the three-shift and two-shift systems, respectively. In each system, each nurse is assigned to exactly one shift per day:

$$\sum_{s \in S^{(3)}} x_{its} = 1, \qquad \sum_{s \in S^{(2)}} y_{its} = 1 \qquad \forall i \in N,\ t \in T.$$

To suppress undesirable shift sequences across different systems near transition boundaries, we introduce a slack variable $\alpha_{itss'}$ and impose the following constraint:

$$x_{i,t-1,s} + y_{i,t,s'} \leq 1 + \alpha_{itss'}, \qquad \forall i \in N,\ t \in T,\ (s, s') \in Q,$$

where Q is the set of forbidden shift pairs. The slack variable $\alpha_{itss'}$ is penalized in the objective function, with weights w_t derived from the first-passage probability.

Additional constraints include staffing requirements, adjusted according to estimated infection-related absences, nurse preferences, workload limits, and off-duty consistency. Staffing requirements and consecutive-work limits are defined separately for the three-shift and two-shift systems based on Japanese hospital staffing practice. Soft constraints include transition-related forbidden patterns, excessive consecutive work near transition periods, and selected staffing requirements.

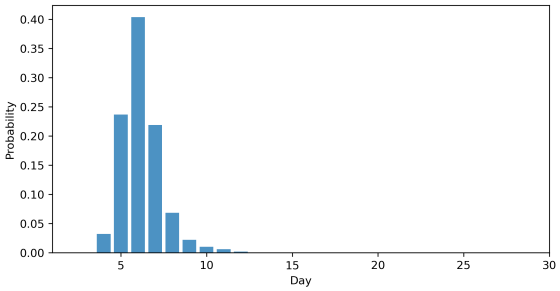


Fig. 1. Probability distribution of the first threshold-exceeding day.

Nurse	Type	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9
Nurse 1	3-shift	O	O	D	S	E	N	O
Nurse 1	2-shift	P	O	D	D	O	N	P
Nurse 2	3-shift	O	D	D	D	N	O	O
Nurse 2	2-shift	O	O	D	D	N	P	O

D : Day E : Evening N : Night P : Post-night O : Off S : Special duty

Fig. 2. Duty roster generated by the proposed method around the transition period.

3 Numerical Experiments

The infection prediction model was calibrated using daily COVID-19 data in Japan. We considered a ward with 25 nurses and generated both three-shift and two-shift duty rosters over a one-month scheduling horizon. The critical threshold for the infection probability was set to 5%.

To evaluate the performance of the proposed method, we compared it with a conventional method under the same set of constraints. In the conventional method, a three-shift roster is first generated, and then re-optimization is performed to make a transition to a two-shift roster, where the transition day is selected from the days with high first-passage probabilities. All models were solved using Gurobi 11.0.3 with a 3600-second time limit on a computer with an Intel Core i5-11320H processor (3.20 GHz).

Fig. 1 shows the first-passage probability of each day, that is, the probability distribution of the day on which the infection probability first exceeds the threshold. The probability is highest around days 5 to 7, indicating that the likelihood of requiring a shift system transition is greatest during this period.

Fig. 2 shows the duty roster generated by the proposed method around the transition period. This figure illustrates that days off in the three-shift schedule correspond to either days off or post-night shifts in the two-shift schedule throughout the scheduling period. Furthermore, during the transition from the three-shift schedule to the two-shift schedule around the candidate transition days (5–7), no forbidden shift patterns or excessive consecutive work are observed, indicating that feasible and consistent shift sequences are maintained.

Table 1. Detailed comparison of the proposed and conventional methods.

	Proposed	Conventional		
Metric		Day 5	Day 6	Day 7
Forbidden pattern violations	0	0	0	0
Consecutive work violations	0	2	2	2
Holiday mismatch	1	0	0	0
Preference violations	0	0	0	0
Night shift lower bound violations	0	11	12	0
Day shift lower bound violations	1	1	1	1
Optimality gap (%)	0	0	0	0
Computation time (seconds)	2100	14	18	20

Table 1 summarizes the quantitative comparison between the proposed and conventional methods. For the conventional method, re-optimization was performed with the transition day set to days 5, 6, and 7, on which the first-passage probabilities are high. Although the proposed method requires longer computation time than the conventional method, it produces more stable schedules with fewer constraint violations, including no consecutive work violations and significantly fewer night-shift lower-bound violations.

4 Conclusion

We proposed a nurse rostering method that incorporates infection-risk-based transitions from a three-shift system to a two-shift system. In conventional approaches, rescheduling after determining the transition timing may make it difficult to maintain consistency between the pre- and post-transition rosters and to eliminate constraint violations. As a result, operational confusion may arise among staff. In contrast, by distributing both the two-shift and three-shift rosters to nurses in advance, the transition can be executed more smoothly, which is expected to mitigate such confusion in practice. Numerical experiments demonstrated that the proposed method improves roster stability around potential transition periods by reducing undesirable shift patterns and consecutive work violations.

Future work includes real-world validation, absence modeling, and better alignment between the two rosters.

References

1. Wang, Y.-Y., Zhang, W.-W., Lu, Z.-X., Sun, J.-L., Jing, M.-X.: Optimal resource allocation model for COVID-19: A systematic review and meta-analysis. BMC Infectious Diseases **24**(1), 200 (2024)
2. Gao, X., Jiang, L., Hu, Y., Li, L., Hou, L.: Nurses' experiences regarding shift patterns in isolation wards during the COVID-19 pandemic in China: A qualitative study. Journal of Clinical Nursing **29**, 4270–4280 (2020)

A Three-Phase Framework for Course Planning and Timetabling in Upper Secondary Schools with Self-Directed Learning

Michael Hölscher[0000−0003−0989−5528]

Chair of Business Administration, esp. Industrial Management, Faculty of Business and Economics, TU Dresden, 01062 Dresden, Germany
michael.hoelscher@tu-dresden.de

Keywords: upper secondary school timetabling · course planning · student sectioning · self-directed learning

1 Introduction

Educational systems show a growing trend towards more individualized learning paths. Particularly in upper secondary education, students can often select elective subjects according to their interests and academic goals instead of following strictly predefined curricula. At the same time, pedagogical concepts increasingly incorporate flexible learning formats alongside traditional classroom teaching. One example is the introduction of self-directed learning sessions (SDL sessions), in which students work independently on subject-specific tasks while teachers provide guidance. The setting considered in this work originates from a research project conducted in collaboration with a small network of upper secondary schools in Germany that integrate SDL sessions into their instructional concept. In these schools, SDL sessions allow students to complete part of their required subject hours through supervised independent learning in addition to regular teacher-led classroom instruction. The combination of individualized course selection and SDL sessions introduces additional combinatorial challenges for timetable planning, which are difficult to handle with existing timetabling software systems and therefore motivate the development of new planning approaches.

2 Problem Description and Related Work

In upper secondary schools in Germany, students can choose from a set of advanced and basic courses when composing their individual curriculum. Advanced courses require a higher weekly workload than basic courses, and each student usually selects two or three advanced courses (depending on the federal state) in addition to several basic courses, resulting in a curriculum of roughly twelve subjects. For each subject, a required number of weekly hours must be fulfilled. In the schools considered in this study, these hours are delivered through a hybrid instructional concept that combines regular classroom lessons with the SDL

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

sessions. Regular lessons follow the traditional format of teacher-led instruction for a group of students, while SDL sessions allow students to work independently on subject-related tasks under the supervision of a teacher. At the same time, the planning process must account for operational constraints such as class size limits, teacher availability, room capacities, and the required number of weekly teaching hours for each subject.

In the classical high school timetabling problem the set of lessons, including their teachers and classes, is fixed in advance and the curriculum is predefined. The task is to assign each lesson or event to a period and a room without creating resource conflicts [7]. Another line of research focuses on university course timetabling and distinguishes between Post-Enrolment Course Timetabling (PE-CTT) [5], where individual student enrollments are known prior to the timetabling process, and Curriculum-Based Course Timetabling (CB-CTT) [2], where predefined curricula represent groups of students that must attend the same set of courses. When course capacities are limited, students must additionally be assigned to specific course sections while avoiding timetable conflicts, a problem commonly referred to as student sectioning [6]. For upper secondary schools with elective systems, Kristiansen et al. introduced the Elective Course Planning Problem (ECPP), which assigns course classes to predefined timetable blocks while maximizing the number of satisfied course requests [4]. Hoshino and Fabris [3] investigate the optimization of student course preferences in school timetabling and consider the construction of compatible course bundles. Furthermore, Aslan et al. [1] investigate learning activity planning in personalized, demand-driven learning environments using a hyper-heuristic framework.

The problem studied in this paper lies between classical school timetabling and university course timetabling due to the elective course structure, while also incorporating student sectioning aspects and elements of personalized learning through flexible SDL sessions.

3 Solution Approach

To address the problem, the following three-phase planning framework is proposed in which the overall planning task is decomposed into separate planning stages:

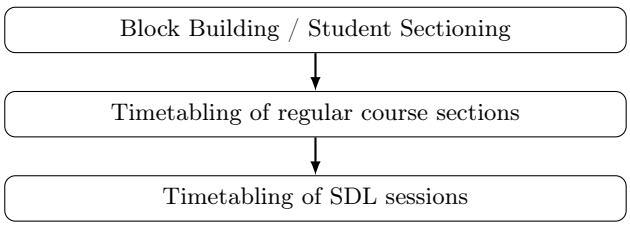


Fig. 1. Three-Phase Framework

Phase 1: Block Building / Student Sectioning In the first phase, a block structure is constructed based on student course selections. A block represents a scheduling unit used in the later timetabling phase and contains a set of course sections that can take place in parallel without creating conflicts for any student. This phase can be interpreted as a combination of course bundling and student sectioning and can be viewed as a conflict-grouping problem in which course sections that share students must not be assigned to the same block. The model determines both the assignment of students to course sections and the assignment of course sections to blocks. Constraints ensure that each student is assigned to a section of every selected course, section capacities are respected, and conflicting sections are placed in different blocks. The objective is to construct a compact block structure with as few blocks as possible, since each block must subsequently be scheduled in the timetable according to the weekly duration of the corresponding courses.

Phase 2: Timetabling of regular course sections In the second phase, the blocks determined in Phase 1 are scheduled within the weekly timetable. Each block is assigned to time slots as often as required to cover the portion of subject hours delivered through regular teacher-led lessons within the hybrid instructional concept, while the remaining hours are fulfilled through SDL sessions. Since the blocks already contain conflict-free course sections determined in Phase 1, the timetabling process operates on the level of blocks rather than individual courses. The model determines the assignment of blocks, teachers, and rooms to time slots. Constraints ensure that all required lesson hours are scheduled while preventing teacher and room conflicts and respecting teacher availability. Additional constraints can be used to enforce pedagogical and organizational requirements. The objective is to produce compact and practical timetables for both students and teachers while ensuring the feasibility of all scheduled lessons.

Phase 3: Timetabling of SDL sessions In the third phase, the timetable constructed for the regular course blocks is complemented by SDL sessions. In this step, it is determined which SDL sessions are offered and at which time slots they take place. Each session is supervised by a teacher and associated with a subject area that determines the subjects for which students can complete learning tasks during that session. The model determines the set of SDL sessions to be offered as well as their assignment to teachers and time slots. Constraints ensure sufficient SDL capacity for all subjects, fulfillment of the remaining subject-specific hour requirements of all students, and compatibility with teacher availability and the timetable generated in Phase 2. The objective of this phase is to ensure that all students can fulfill the remaining required subject hours not covered by regular classroom lessons. SDL sessions are therefore scheduled while remaining compatible with the previously constructed timetable and giving preference to time slots with the greatest potential to satisfy outstanding student demand.

The solution framework is implemented through three separate mathematical optimization models corresponding to the three planning phases. This decomposition reduces the size and complexity of the individual models and limits the combinatorial interactions between decisions occurring at different planning levels. In contrast, a fully integrated formulation would lead to a substantially larger and more complex model that is difficult to solve in practice. The phases are currently solved sequentially, with the output of each phase serving as input to the subsequent phase. While this decomposition may exclude some globally optimal solutions, it substantially improves tractability and enables the solution of practically relevant instances. To evaluate the computational performance of the approach, implementations based on the Gurobi solver as well as a CP-SAT solver were developed.

Initial computational experiments on representative real-world instances indicate that all three phases can be solved within acceptable computation times. Further computational studies are currently being conducted to systematically assess the performance, scalability, and solution quality of the proposed decomposition approach, including the potential benefits of feedback mechanisms between planning phases. The overall goal is to establish the framework as a practical decision-support tool for schools implementing flexible pedagogical concepts and enabling students to pursue more individualized learning paths.

Acknowledgements This research was conducted within a project funded by the German Federal Ministry for Education, Family Affairs, Senior Citizens, Women and Youth under grant number 01JQ2403A-B.

References

1. Aslan, A., Bakir, I., Vis, I. F. (2020). A dynamic thompson sampling hyper-heuristic framework for learning activity planning in personalized learning. European Journal of Operational Research, **286**(2), 673–688.
2. Di Gaspero, L., McCollum, B., Schaerf, A. (2007). The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3). Technical Report, Queen's University, Belfast, United Kingdom.
3. Hoshino, R., Fabris, I. (2020). Optimizing student course preferences in school timetabling. In International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research, 283–299.
4. Kristiansen, S., Sørensen, M., Stidsen, T. (2011). Elective course planning. European Journal of Operational Research, **215**, 713–720.
5. Lewis, R., Paechter, B., McCollum, B. (2007). Post enrolment based course timetabling: A description of the problem model used for track two of the second international timetabling competition. Technical Report, Cardiff University, Wales, United Kingdom.
6. Müller, T., Murray, K. (2010). Comprehensive approach to student sectioning. Annals of Operations Research, **181**(1), 249–269.
7. Schaerf, A. (1999). A survey of automated timetabling. Artificial Intelligence Review **13**, 87–127.

Smart TV Scheduling in Public Venues: A New Combinatorial Problem and Optimization Approaches

Shefket Bylygbashi, Fisnik Hazrolli, Endrita Vllasaliu, Eljon Shala, Yll Berisha, Kaltrina Krasniqi, Kadri Sylejmani, Labeat Arbneshi, and Uran Lajci

Faculty of Electrical and Computer Engineering,
University of Prishtina, Prishtina, Kosovo
{shefket.bylygbashi, fisnik.hazrolli, endrita.vllasaliu, eljon.shala, yll.berisha3, kaltrina.krasniqi31}@student.uni-pr.edu
{kadri.sylejmani, labeat.arbneshi, uran.lajqi}@uni-pr.edu

Abstract. We introduce the *Smart TV Scheduling in Public Venues* problem, a new combinatorial optimization problem in which an intelligent system must select and schedule content from multiple TV channels on a single screen in a public venue to maximize aggregate viewer satisfaction, allowing gaps in the schedule, subject to operational constraints including genre diversity, minimum continuity, priority time blocks, and channel-switch and partial-broadcast penalties. We propose a hybrid pipeline combining a Genetic Algorithm (GA) seeded from the better of a Beam Search and a greedy segment-based initial solution, a Local Search refinement phase (GA+LS), and a CP-SAT exact solver warm-started from the best GA+LS solution. Experiments on 16 benchmark instances covering traditional TV broadcast, IPTV, provider-based, and YouTube scenarios demonstrate that GA+LS consistently produces high-quality incumbents that enable CP-SAT to confirm optimality on all synthetic instances and to achieve competitive scores on the larger real-life ones within a 3,600-second time limit. The results highlight that the strength of the pipeline lies in the complementarity of its components: GA+LS provides fast heuristic exploration, while CP-SAT leverages the resulting incumbent to deliver the refinement and optimality guarantees that a pure heuristic cannot offer.

Keywords: TV Scheduling · Genetic Algorithm · Local Search · Constraint Programming · Metaheuristics

1 Problem Definition

The *Smart TV Scheduling in Public Venues* problem tasks an intelligent system with selecting and displaying content from multiple available TV channels on a single screen inside a public venue. The objective is to construct a broadcast

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

schedule over the venue's operating hours that maximizes aggregate viewer satisfaction while navigating complex operational constraints and penalties. Gaps between scheduled programs are permitted but earn no score.

Inputs and Outputs. A problem instance specifies the venue's opening and closing times and a set of available channels, each broadcasting a fixed sequence of non-overlapping programs. Every program is characterized by a start time, an end time, a genre, and a popularity score. The system must output a valid schedule as an ordered list of tuples $(\text{channel ID}, \text{program ID}, s, e)$, where s and e denote the actual displayed start and end times of the selected program, respectively. This representation explicitly captures the broadcast sequence as well as any partial broadcasts or gaps between consecutive programs. A channel may appear multiple times, but each program is scheduled at most once. Since program times are fixed in the input, any necessary partial broadcasts are implicitly derived from the sequencing of the output list.

Operational Constraints. To be considered valid, the schedule must strictly adhere to the following rules:

- **Temporal Feasibility:** All scheduled programs must start no earlier than the opening time and end no later than the closing time.
- **Non-overlap:** Scheduled programs cannot overlap in their actual displayed timelines.
- **Genre Diversity:** The sequence cannot contain more than R consecutive programs sharing the same genre.
- **Minimum Continuity:** Every scheduled program must be broadcast continuously for at least D minutes. If a program's total duration is shorter than D, it must be scheduled for its full original duration.
- **Priority Compliance:** During designated priority time blocks, only programs from a specified subset of allowed channels may be scheduled.

Objective Function. The system aims to maximize a net score defined as:

$$\begin{aligned}\text{Total Score} = &\sum \text{program scores} + \sum \text{time-block bonuses} \\ &- S \times (\text{channel switches}) - T \times (\text{partial broadcasts})\end{aligned}$$

where S is the per-switch penalty and T is the per-partial-broadcast penalty. A time-block bonus is awarded when a scheduled program's genre matches a preferred genre interval and the program overlaps that interval by at least D minutes. The penalty T applies both when a program is terminated before its official end time and when it is joined after its official start time.

1.1 Mathematical Model

We now give a precise formulation of the problem. This same model is solved exactly by the constraint-programming (CP-SAT) baseline used in our experiments (Section 3). We state it in the native form used by the solver: variables may take integer or Boolean values, and a constraint written "if A then B" (denoted $A \Rightarrow B$) is enforced directly through reification rather than through big-M linearization.

Sets and parameters. Let P be the set of candidate programs (those whose original window overlaps the operating day), C the channels, G the genres, B the priority blocks, and W the time preferences. The day runs over $[O, E]$ in minutes. The constants D, R, S, and T are the minimum airing duration, the maximum allowed same-genre streak, the per-switch penalty, and the per-partial-broadcast penalty, respectively. Each program $i \in P$ has channel $c_i \in C$, genre $g_i \in G$, popularity score v_i, original window $[a_i, b_i)$, and original duration $\delta_i = b_i - a_i$. Each priority block $b \in B$ has a window $[\alpha_b, \beta_b)$ and an allowed-channel set $A_b \subseteq C$. Each preference $w \in W$ has a window $[\alpha_w, \beta_w)$, a target genre $\hat{g}_w$, and a bonus ρ_w; we write W_i for the preferences compatible with program i, i.e. those with $\hat{g}_w = g_i$ whose window can overlap i by at least D minutes.

Decision variables. For every program $i \in P$:

- $x_i \in \{0, 1\}$: whether program i is included in the schedule;
- $s_i, e_i \in [\max(O, a_i), \min(E, b_i)]$: the actual displayed start and end time, with chosen duration $d_i = e_i - s_i \in [0, \delta_i]$;
- $\sigma_i^{\text{late}}, \sigma_i^{\text{early}} \in \{0, 1\}$: indicators for a late join and an early termination;
- $z_{iw} \in \{0, 1\}$ for each $w \in W_i$: whether i earns preference bonus w;
- $k_i \in \{1, \ldots, R\}$: the length of the same-genre streak ending at i.

For sequencing we also use arc variables $\text{succ}_{ij} \in \{0, 1\}$ over the node set $P \cup \{\text{src}, \text{snk}\}$, where $\text{succ}_{ij} = 1$ means program j airs immediately after program i.

Minimum continuity. A scheduled program must air for at least D minutes, or for its whole length if it is shorter than D; an unscheduled program has zero duration:

$$d_i \geq \min(D, \delta_i) \;\text{ if } x_i = 1, \qquad d_i = 0 \;\text{ if } x_i = 0, \qquad \forall i \in P. \tag{1}$$

Non-overlap. The displayed intervals of any two scheduled programs may not overlap. This is posted directly as a global non-overlap constraint over the optional intervals $[s_i, e_i)$ of the present programs:

$$x_i = 1 \;\wedge\; x_j = 1 \;\Rightarrow\; e_i \leq s_j \;\vee\; e_j \leq s_i, \qquad \forall\, i \neq j. \tag{2}$$

This is implied by the sequencing constraints below, but is stated explicitly to strengthen propagation.

Priority compliance. Inside a priority block, a program whose channel is not allowed must either finish before the block opens or start after it closes:

$$x_i = 1 \;\Rightarrow\; \big(e_i \leq \alpha_b\big) \;\vee\; \big(s_i \geq \beta_b\big), \qquad \forall b \in B,\;\; \forall i \in P : c_i \notin A_b. \tag{3}$$

Partial-broadcast indicators. A late-join indicator is on exactly when a present program starts after its official start, and an early-termination indicator is on exactly when a present program ends before its official end:

$$\sigma_i^{\text{late}} = 1 \;\Leftrightarrow\; \big(x_i = 1 \;\wedge\; s_i > a_i\big), \qquad \forall i \in P, \tag{4}$$

$$\sigma_i^{\text{early}} = 1 \;\Leftrightarrow\; \big(x_i = 1 \;\wedge\; e_i < b_i\big), \qquad \forall i \in P. \tag{5}$$

Each program can therefore incur up to two partial-broadcast penalties, one at each end.

Time-preference bonuses. A program earns a compatible bonus only if it is scheduled and overlaps the preference window by at least D minutes:

$$z_{iw} = 1 \;\Rightarrow\; x_i = 1 \;\wedge\; s_i \le \beta_w - D \;\wedge\; e_i \ge \alpha_w + D, \qquad \forall i \in P,\;\; \forall w \in W_i. \tag{6}$$

Sequencing. The scheduled programs must form a single ordered broadcast sequence. A directed circuit is built over $P \cup \{\text{src}, \text{snk}\}$ with the closing arc $\text{snk} \to \text{src}$ forced on; a self-loop $i \to i$ is taken exactly when i is *not* scheduled, so that the circuit visits precisely the chosen programs. Whenever j follows i, program i must finish no later than j starts:

$$\text{succ}_{ij} = 1 \;\Rightarrow\; e_i \le s_j, \qquad \forall\, i, j \in P,\; i \ne j. \tag{7}$$

Because every scheduled program has positive duration, this ordering forces start times to strictly increase along the chain and rules out subtours.

Genre diversity. A streak counter runs along the sequence: it grows by one when consecutive programs share a genre and resets to one otherwise. The first program in the sequence starts a streak of length one, and no counter may exceed R:

$$\text{succ}_{ij} = 1 \;\Rightarrow\; k_j = \begin{cases} k_i + 1, & \text{if } g_i = g_j, \\ 1, & \text{otherwise,} \end{cases} \qquad 1 \le k_j \le R. \tag{8}$$

Objective. Let a *switch* be an active arc joining two different channels. The model maximizes the total program score plus earned bonuses, minus the switch and partial-broadcast penalties:

$$\max \;\sum_{i \in P} v_i\, x_i \;+\; \sum_{i \in P} \sum_{w \in W_i} \rho_w\, z_{iw} \;-\; S \sum_{\substack{i,j \in P \\ c_i \ne c_j}} \text{succ}_{ij} \;-\; T \sum_{i \in P} \big(\sigma_i^{\text{late}} + \sigma_i^{\text{early}}\big). \tag{9}$$

This is exactly the net score of Figure 1: program scores and time-block bonuses are rewarded, while each channel switch costs S and each clipped end costs T.

Illustrative Example. Figure 1 illustrates a toy instance with 4 channels over a 90-minute window (D=10, R=2, S=1, T=2). Panel (a) shows all available programs across channels; selected programs are highlighted while unselected ones are faded. A priority block from minute 20 to 60 restricts broadcasting to

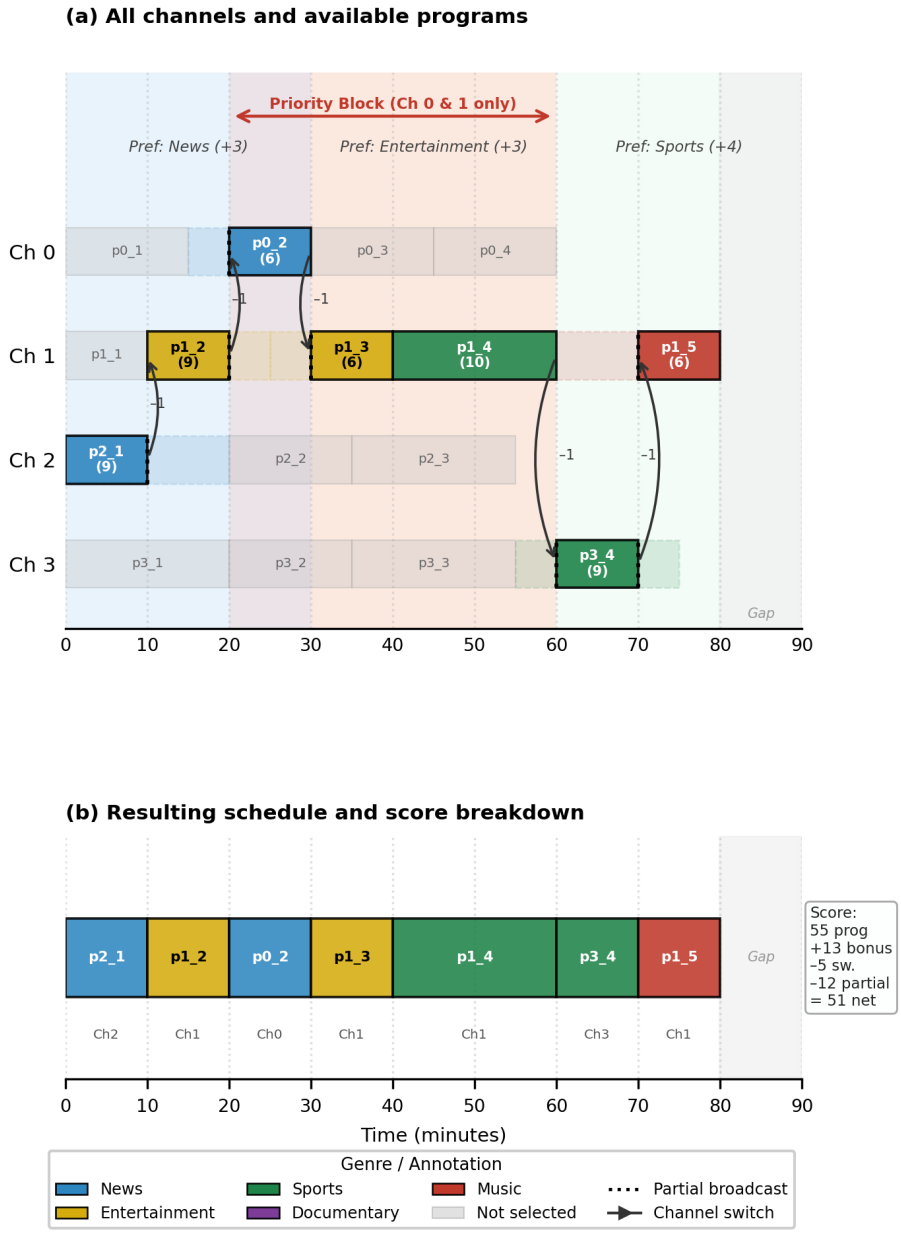


Fig. 1. Illustrative example of the *Smart TV Scheduling in Public Venues* problem on a toy instance (D=10, R=2, S=1, T=2).

Channels 0 and 1, and three time-preference intervals reward news, entertainment, and sports content in successive windows. The resulting schedule shown in panel (b) selects seven programs with a total program score of 55, earning 13 bonus points from time-preference matches. Five channel switches incur a penalty of -5 and six partial broadcasts (programs joined late or terminated early) incur a further penalty of -12, yielding a net score of 51.

2 Solution Approach

We propose a two-phase heuristic pipeline: a *Genetic Algorithm* (GA) that searches for high-quality feasible schedules, followed by a *Local Search* refinement phase (GA+LS) that improves the best solution found. The best GA+LS solution is then passed as a warm-start to the CP-SAT exact solver. Source code for all components is publicly available on GitHub (see Section 3).

Genetic Algorithm (GA). The GA begins by constructing a deterministic initial solution by running both a Beam Search and a segment-based deterministic approach [9,3] and seeding the population with the better of the two solutions.[1] A population of candidate schedules is then created as variants of this seed and evolved over multiple generations. In each generation, the population is ranked by score; the best individuals are carried forward unchanged through elitism; parents are selected via tournament selection; offspring are produced by crossover and may be further perturbed by mutation; and every offspring is repaired to restore feasibility before being evaluated.

Two crossover operators are used. *Time-cut crossover* selects a cut point in time and combines the portion before that point from one parent with the portion after it from the other, exploiting the temporal structure of the schedule. *Block-mix crossover* selects a time interval and takes a block of segments from one parent while the remainder comes from the other, preserving high-quality sub-schedules across recombination.

Five mutation operators are applied with varying disruption: *Replace segment* substitutes an existing segment with a nearby alternative from the candidate pool; *Delete weak* removes the segment with the lowest marginal contribution to the score; *Insert gap* finds an empty interval and attempts to fill it with a new program; *Expand boundary* widens a partial broadcast to a longer valid version of the same program, reducing partial-broadcast penalties; and *Block rebuild* removes a contiguous portion of the schedule and reconstructs it from scratch, enabling stronger exploration. If no improvement is observed for several generations, a more disruptive mutation is applied to escape local optima.

After crossover or mutation, a repair procedure removes invalid segments, resolves overlaps, eliminates duplicate program uses, corrects genre-streak violations, and attempts to fill resulting gaps, returning the schedule to a fully feasible

[1] Earlier stages of this project also explored a Beam Search combined with Simulated Annealing pipeline [4] as a standalone solver. Those results served as a proof of concept and are superseded by the GA+LS pipeline presented here.

state. Each individual is then scored according to the objective function: program scores and time-preference bonuses are rewarded, while channel switches and partial broadcasts are penalised. The key tuned parameters are population size, elite count, and top-K (the number of candidate segments retained per program).

Local Search (GA+LS). After the GA phase, a multi-neighbourhood greedy local search is applied to the best solution found. Rather than restarting from scratch, it explores the neighbourhood of the incumbent through several move types: replacing segments with time-adjacent alternatives, inserting programs into free slots, expanding partial broadcasts, removing low-contribution segments, rebuilding small blocks, and extending runs on the same channel to reduce switch penalties. Each candidate move is repaired and re-evaluated; only improving moves are accepted. The GA thus handles broad exploration of the solution space, while the local search provides fine-grained intensification around the most promising solution found.

3 Computational Results

All experiments were conducted on a standard desktop machine. The 16 benchmark instances span four scenario types – traditional TV broadcast (TV), internet-based (IPTV), provider-based (PW), and YouTube – covering a range of channel counts, program densities, and constraint configurations, and represent a subset of the full benchmark dataset introduced in [8].

Table 1 summarizes the 16 benchmark instances and their key structural parameters. They span more than three orders of magnitude in program count, from the compact *germany_tv* instance with 52 programs to the very large *us_iptv* instance with 115,446 programs. The synthetic TV instances use shorter horizons (900–1,080 minutes) and few time-preference intervals (5–6), whereas the real-life-based IPTV, PW, and YouTube instances run over a full day or longer and include up to 26 such intervals. Although the average number of programs per channel stays within a relatively narrow band (about 10–20), the instances differ sharply in constraint structure: the smaller instances are more tightly constrained – with priority blocks admitting only a handful of channels – while the larger instances are looser but combinatorially far more complex. Per-instance values of the constants D, R, S, and T are provided with the dataset [8].

Experimental setup. The GA was run 10 times per instance with a budget of approximately 5 minutes per run (roughly 50 minutes of total computation per instance), varying key parameters – including population size, elite count, and top-K selection – across runs; the configuration yielding the highest score was then fixed and used as the reported result. For GA+LS, the best parameter configuration identified from the GA runs was adopted and fixed; each run then executes the GA phase for approximately 60 seconds, followed immediately by 60 seconds of local search, for a total of roughly 2 minutes per run. GA+LS was also repeated 10 times per instance, with the best solution across all runs used as the warm-start for CP-SAT. Each instance was then solved with a 3,600-second

Table 1. Characteristics of the 16 benchmark instances. "Type" marks whether the instance is synthetic or derived from real-life data; "Hor." is the operating horizon in minutes; "Prog./Ch." the average programs per channel; "Gen.", "Pref.", and "Blocks" the numbers of genres, time-preference intervals, and priority blocks. The leading # matches the numbering of Table 2.

#	Instance	Type	Hor.	Chan.	Prog.	Prog./Ch.	Gen.	Pref.	Blocks
1	croatia_tv	Synth.	1,020	15	205	13.67	10	5	2
2	germany_tv	Synth.	900	5	52	10.40	15	5	2
3	kosovo_tv	Synth.	960	13	175	13.46	8	5	2
4	netherlands_tv	Synth.	1,080	12	180	15.00	10	6	2
5	uk_tv	Synth.	900	120	1,621	13.51	10	5	2
6	usa_tv	Synth.	1,080	1,100	16,277	14.80	10	5	2
7	australia_iptv	Real	1,440	499	8,151	16.33	13	26	3
8	canada_pw	Real	1,440	634	12,162	19.18	13	26	3
9	china_pw	Real	1,440	1,254	20,429	16.29	8	26	3
10	france_iptv	Real	1,440	397	6,291	15.85	13	26	2
11	singapore_pw	Real	1,440	211	4,158	19.71	13	26	2
12	spain_iptv	Real	1,440	269	4,448	16.54	13	26	2
13	uk_iptv	Real	1,440	903	12,441	13.78	13	26	2
14	us_iptv	Real	1,440	11,457	115,446	10.08	13	26	3
15	youtube_gold	Real	20,160	1,422	25,198	17.72	12	12	6
16	youtube_premium	Real	21,600	1,677	29,590	17.64	7	12	6

CP-SAT time limit. If the solver proved optimality within this limit, the result is reported as *Opt.*; if the time limit was reached before optimality could be proven but a feasible solution was found, the result is reported as *Feas.*, corresponding to the best incumbent found within the computational budget without a global optimality guarantee.

All solvers and tools developed in this work are publicly available on GitHub, including the Beam Search and greedy initial solution implementations [9], the Segment-Based Deterministic Scheduler [3], the Genetic Algorithm and Local Search pipeline [2], the CP-SAT exact solver [1], and two web-based schedule validation tools [7,5].

Table 2 reports scores for GA, GA+LS, and CP-SAT across all 16 instances. The central motivation for warm-starting CP-SAT from GA+LS is that CP-SAT, like all exact solvers, explores the solution space by progressively tightening bounds – a process that on large instances can consume the entire time budget before reaching a good feasible solution. By injecting a high-quality GA+LS solution at the start, the solver immediately has a strong lower bound to prune against, which drastically reduces the effective search space and allows CP-SAT to spend its time budget on refinement rather than on finding a first feasible solution from scratch. This is particularly important on instances such as *us_iptv* (115,446 programs) or *youtube_premium* (29,590 programs), where the combinatorial space is so large that a cold-started CP-SAT would be unlikely to find any competitive solution within the 3,600-second limit.

On the six synthetic TV instances, CP-SAT reaches optimality in all cases; in five of these six the warm-start value provided by GA+LS already equals the final CP-SAT score, meaning the solver confirms optimality without any further improvement – a direct consequence of GA+LS delivering near-optimal

Table 2. Results across 16 instances. CP-SAT is warm-started from the best GA+LS solution across 10 runs; Impr. shows the percentage gain of CP-SAT Final over the warm-start value. St.: solver status (Opt. = proven optimal within the time limit, Feas. = best incumbent within the time limit).

			GA		GA+LS		CP-SAT		
#	Instance	St.	Avg	Max	Avg	Max	Warm	Final	Impr. (%)
1	croatia_tv	Opt.	2,220.0	2,220	2,220.0	2,220	2,220	2,220	0.00
2	germany_tv	Opt.	1,604.0	1,608	1,604.0	1,608	1,608	1,608	0.00
3	kosovo_tv	Opt.	2,588.4	2,591	2,590.6	2,591	2,591	2,591	0.00
4	netherlands_tv	Opt.	2,634.6	2,635	2,633.8	2,635	2,635	2,635	0.00
5	uk_tv	Opt.	2,266.0	2,266	2,266.0	2,266	2,266	2,266	0.00
6	usa_tv	Opt.	3,598.1	3,601	3,598.0	3,598	3,598	3,601	0.08
7	austr._iptv	Feas.	4,886.7	4,897	4,888.2	4,899	4,899	4,903	0.08
8	canada_pw	Feas.	5,668.6	5,671	5,670.2	5,671	5,671	5,671	0.00
9	china_pw	Feas.	3,056.1	3,116	3,054.3	3,083	3,083	3,136	1.72
10	france_iptv	Feas.	10,983.0	10,983	10,983.0	10,983	10,983	10,998	0.14
11	singap._pw	Feas.	6,986.0	6,986	6,986.0	6,986	6,986	6,987	0.01
12	spain_iptv	Feas.	6,655.0	6,655	6,655.0	6,655	6,655	6,655	0.00
13	uk_iptv	Feas.	9,948.0	9,948	9,948.2	9,950	9,950	9,950	0.00
14	us_iptv	Feas.	5,560.0	5,560	5,560.0	5,560	5,560	5,560	0.00
15	yt._gold	Feas.	107,436.2	107,439	107,441.1	107,452	107,452	107,452	0.00
16	yt._prem.	Feas.	67,862.0	67,862	67,862.0	67,862	67,862	67,862	0.00

solutions on these smaller, well-constrained instances. On the ten larger real-life instances (IPTV, PW, and YouTube), CP-SAT remains feasible within the time limit, and the benefit of warm-starting is most visible: the solver begins from a strong incumbent and uses its remaining budget to close the gap further, yielding additional gains of 1.72% on *china_pw* and 0.14% on *france_iptv*. GA and GA+LS produce near-identical results across most instances, suggesting that the local search phase reliably reaches high-quality solutions that serve as effective starting points. Overall, the pipeline is designed so that each component compensates for the weakness of the other: GA+LS explores the space quickly and heuristically, providing CP-SAT with a strong incumbent, while CP-SAT leverages that incumbent to provide final refinement and, when optimality is proven within the time limit, global optimality guarantees; otherwise, it returns the best feasible incumbent found within the available computational budget.

4 Conclusion and Future Work

We introduced the *Smart TV Scheduling in Public Venues* problem and proposed a hybrid pipeline combining a Genetic Algorithm (GA) seeded from the better of a Beam Search and a greedy segment-based initial solution, a Local Search refinement phase (GA+LS), and a CP-SAT exact solver warm-started from the best GA+LS incumbent. Experiments on 16 benchmark instances spanning synthetic TV and real-life IPTV, provider-based, and YouTube scenarios demonstrate that GA+LS consistently produces high-quality solutions that enable CP-SAT to confirm optimality on all six synthetic instances and to achieve competitive scores on the ten larger real-life instances within a 3,600-second time

limit. The results highlight that neither a pure heuristic nor a standalone exact solver is sufficient at this scale: the strength of the pipeline lies in the complementarity of its components, with GA+LS providing fast heuristic exploration and CP-SAT leveraging the resulting incumbent to deliver the refinement and optimality guarantees that a heuristic alone cannot offer.

Future work will explore alternative metaheuristic strategies – such as ant colony optimization and adaptive large neighborhood search with learned operator weights [6] – as warm-start providers for CP-SAT, to investigate whether stronger incumbents can further reduce the gap on the hardest instances such as *china_pw*, where CP-SAT still improved upon the GA+LS warm-start by 1.72%. Extensive experimentation on the full benchmark dataset [8] beyond the 16 instances studied here also remains an important direction, as does the investigation of decomposition strategies that could make the exact CP-SAT formulation more tractable on very large instances such as *us_iptv* (115,446 programs).

References

1. Berisha, Y.: Smart TV scheduling – CP-SAT solver with OR-Tools. `https://github.com/yllberisha/tv-scheduling-ortools` (2025), accessed: 2025
2. Bylygbashi, S.: Smart TV scheduling – genetic algorithm solver. `https://github.com/shefketbylykbashi/smart-tv-scheduling-ga` (2025), accessed: 2025
3. Bylygbashi, S.: Smart tv scheduling: Segment-based deterministic scheduler (sbds) (2025), `https://github.com/shefketbylykbashi/smart-tv-scheduling-sbds`. University of Prishtina, Faculty of Electrical and Computer Engineering
4. Hazrolli, F., Vllasaliu, E., et al.: Metaheuristic tv scheduling: Implementation of beam search and simulated annealing for smart tv scheduling in public venues (2025), `https://github.com/fisnikhz/metaheuristic-tv-scheduling`. University of Prishtina, Faculty of Electrical and Computer Engineering
5. Krasniqi, K.: Smart tv scheduling validator (2025), `https://smarttvschedulingvalidator.netlify.app/`. Accessed: March 2026
6. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science **40**(4), 455–472 (2006)
7. Shala, E.: Tv scheduling validator: An online validation tool for the smart tv scheduling optimization problem (2025), `https://tv-programms-validator.netlify.app/`. Accessed: March 2026
8. Sylejmani, K., Bylygbashi, S., Lajçi, U., Kastrati, Z.: A curated dataset of multichannel tv program schedules for optimization and benchmarking. Data in Brief p. 112568 (2026)
9. Vllasaliu, E., Bylygbashi, S., et al.: AA_25-26: Implementation of smart tv scheduling approaches (2025), `https://github.com/endritavllasaliu16/AA_25-26`. Accessed: 2025

Mathematical Optimization of the Rail Infrastructure Planning Timetable

Alexander Heide Madsen[1,2] [0009−0000−0426−6883],
Victor Alexander Brandsen[1,2] [0009−0008−4823−1496], and
Thomas Stidsen[1] [0000−0001−6905−5454]

[1] Technical University of Denmark, 2800 Kgs. Lyngby, Denmark ahema@dtu.dk
[2] Macom A/S, 1620 København, Denmark

Abstract. The long-term timetabling of railway renewal and maintenance is a complex and large-scale problem with significant economical and societal implications. We present a mixed-integer programming model for solving the Danish Rail Renewal and Maintenance Problem (DRRMP), considering the trade-offs between renewal, maintenance and their impact on traffic disturbance.
We consider a time horizon of between 60 and 120 years and the entire Danish railway network. The proposed model incorporates realistic operational constraints provided by Banedanmark, and is evaluated on data provided by Banedanmark, to test its applicability on large-scale infrastructure planning problems.

Keywords: Transport · Strategic planning · Mathematical Programming.

1 The Danish Rail Renewal and Maintenance Problem

Banedanmark (BDK), the Danish railway infrastructure manager, is responsible for producing an annual financial budget that includes a long-term timetable of renewal and maintenance activities. To produce a realistic and reliable budget, a renewal and maintenance timetable must be constructed. This is an extremely intricate and costly process, as the Danish railway infrastructure consists of thousands of rail-segments, sleepers, ballast and switches, spanning over 3000 km. Further details about the network is provided in figure 1. Currently, BDK relies on legacy software to construct the renewal timetable,



Fig. 1. Map of the Danish railway network

and a recent market survey failed to identify suitable alternative software solutions. For this reason, we introduce a complex mathematical model for solving the Danish Rail Renewal and Maintenance Problem (DRRMP) on a strategic level with a time horizon spanning from 60 on 120 years. To the best of our knowledge, no previous studies have considered the same planning horizon on a railway network of this scale. However, related work have addressed similar railway scheduling and planning problems. [1] address the same maintenance planning problem on the Swedish railways and uses a using a matheuristic to construct a day-to-day plan with a time horizon of 10 days. [3] considers an annual planning of 600 fixed maintenance projects of the Dutch railway system using a Hybrid Greedy-ES algorithm, and finally . Furthermore the DRRMP differentiate from the problem posted by [3] considers scheduling predetermined project requests on a much smaller time scale, whereas the DRRMP model considered the creation and scheduling of renewal and maintenance projects on a much larger scale.

1.1 Objective

The purpose of the DRRMP model is to provide BDK with a new state-of-the-art method to construct renewal and maintenance timetables. This model evaluates the trade-offs between renewing and maintaining the railway assets, along with the socioeconomic impact on the system. To this end , the model considers 3 intertwined sub-objectives *Maintenance cost*, *Renewal Cost*, and *Traffic Hindrance Cost*. As all three components are expressed in monetary terms (million DKK), they can be aggregated directly into the single linear objective function shown in (1). This enables the model to balance the direct costs of maintenance and renewal activities against the socioeconomic costs associated with train delays.

Maintenance cost: The cost of maintaining a given asset not only depends on the assets age - generally the cost increases as the asset gets older - but also on the amount of traffic on the asset. Assets located on heavily trafficked sections require more frequent maintenance. The price of maintaining an asset is often cheaper than renewing it. However an asset can only be maintained a certain amount of times before renewal is required.

Renewal cost: The cost of renewing assets depends on the combination of asset renewals. As briefly described earlier, an asset is one of four items, *rails*, *sleepers*, *ballast* or *switches*, where the first three are physically located within the track structure. For this reason, the cost of renewing a combination of assets may sometimes be be cheaper than the sum of the individual renewals.

Traffic Hindrance Cost (THC): The THC represents the socioeconomic impact of the renewal and maintenance timetable. As railway assets deteriorate, the likelihood of speed restrictions and operational disruptions increases, leading to train delays. These delays impose costs on society by increasing travel times for passengers and delaying the transport of goods. Consequently, THC quantifies the socioeconomic consequences associated with reduced infrastructure performance and the resulting train delays.

1.2 Instance

As mentioned, the model will be solving a real-would instance of planning renewal and maintenance over the next 60 to 120 years for the entire Danish railway network, a system consisting of over 3000 km of rails, sleepers, and ballast. BDK has divided the different assets into segments, with the shortest segments being only 10 meters, resulting in the system having more than 24.000 assets. As some assets are physically located on top of each other, they will be bundled into collections called *asset collections*, such that a single asset collection consists of *rails* (steel tracks guiding the train), *sleepers* (transverse supports holding the rails in place) and *ballast* (crushed stone providing stability and drainage) or just *switches and crossings* (track components enabling trains to change from one track to another) . This results in over 9.000 asset collections. The decision is therefore not whether to renew or maintain individual assets, but which renewal and maintenance *activity* should be done on the asset collection. A renewal or maintenance activity may involve a single asset or a combination of assets within the same collection. Furthermore, BDK groups the renewal and maintenance into projects, as they want to ensure some assets are renewed or maintained at the same time as others, and to ensure a somewhat consistent amount of activities planned throughout the year.

The full-scale timetable instance results in a highly combinatorial problem with millions of possible activity assignments. The model must capture interactions between assets, project bundling, and the probabilistic impact of asset deterioration on train delays, which makes the formulation both large and computationally challenging.

1.3 Model Overview

The DRRMP formulation incorporates the operational constraints considered by BDK when constructing a renewal and maintenance timetable. The most important constraints are summarized in Table 1: Important constraints. In addition to replicating the functionality of the current legacy software, the model also introduces several additional constraints that allow planners to better control the structure of renewal projects, listed in 1: Additional constraints.

The core decision variable of the model is the binary variable $X^{t,p}_{ac,k}$. The variable equals 1 iff renewal or maintenance activity k is performed on asset collection ac in time period t as part of project p, and 0 otherwise. These decisions determine the renewal and maintenance schedule of all asset collections over the timetable horizon. From the selected activities, the model derives the resulting asset conditions, project composition and associated costs.

The objective is to minimize the total cost of renewal, maintenance, and traffic hindrance over the timetable horizon:

$$\min \sum_{t} (C^{t}_{renewal} + C^{t}_{maintenance} + C^{t}_{THC}) \tag{1}$$

The resulting formulation is a large-scale mixed-integer program, which when instantiated on a single line section contains over 300.000 variables and 6 million

	Important constraints
Activity limit	At most one activity can be performed on an asset collection in a year.
Renewal pause	Renewing an asset incurs a 10 year renewal pause on all assets on the same line.
Max train delay	An upper bound on the number of trains that are delayed each year.
Max activity volume	An upper bound on the amount of each activity performed each year.
	Additional constraints
Horizontal bundling	Renewal activities must be grouped into contiguous track segments with a minimum project length.
SVAM	Modeling the number of track-closing-days when work is being conducted.
Project stability	The number of projects may only change by a given amount from year to year.
Renewal budget	The renewal cost may not exceed the renewal budget each year.
Maintenance budget	The maintenance cost may not exceed the maintenance budget each year.

Table 1. Key constraints included in the DRRMP model. Not all constraints are included in the table.

constraints. Scaling to the full Danish railway network, consisting of 50 line sections, results in more than 60 million variables and 130 million constraints. This leads to a highly memory-intensive model that exceeds the capabilities of standard personal computers. Consequently, all experiments are conducted on the high-performance computing cluster [2] at the Technical University of Denmark.

2 A brief conclusion

As the project is still in progress, results from solving the model on the entire network and smaller subparts will be presented along with the conclusions drawn.

References

1. Consilvio, A., Febbraro, A.D., Sacco, N.: A rolling-horizon approach for predictive maintenance planning to reduce the risk of rail service disruptions. IEEE Transactions on Reliability **70**(3), 875–886 (2021). https://doi.org/10.1109/TR.2020.3007504
2. DTU Computing Center: DTU Computing Center resources (2024). https://doi.org/10.48714/DTU.HPC.0001

3. Oudshoorn, M., Koppenberg, T., Yorke-Smith, N.: Optimization of annual planned rail maintenance. Computer-Aided Civil and Infrastructure Engineering **37**(6), 669–687 (2022). https://doi.org/https://doi.org/10.1111/mice.12764

Proceedings of the 15th International Conference on the Practice and Theory
of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

The Integrated Healthcare Timetabling Competition 2024: Insights, Benchmarking and Algorithmic Trends

Sara Ceschia[1], Roberto Maria Rosati[2,3], Andrea Schaerf[1], Pieter Smet[4], Greet Vanden Berghe[4], Eugenia Zanazzo[1]

[1] University of Udine, Udine, Italy
[2] WU Vienna University of Economics and Business, Vienna, Austria
[3] University of Klagenfurt, Klagenfurt am Wörthersee, Austria
[4] KU Leuven, Gent, Belgium

Abstract. The Integrated Healthcare Timetable Competition 2024 introduced the Integrated Healthcare Timetabling Problem, which combines patient admission scheduling, surgical case planning and nurse-to-room assignment into a single large-scale optimization problem. The competition attracted 32 teams from both industry and academia, who developed a diverse range of solution approaches, primarily based on mathematical programming, metaheuristics and hybrid methods. All approaches applied some form of decomposition and incorporated tailored strategies to cope with the problem's scale and combinatorial complexity. Notably, different methods performed best on different instances, and no algorithm was able to prove optimality for any instance.
This work resumes the competition setup and analyzes algorithmic trends, highlighting key methodological insights for integrated timetabling problems.

Keywords: Timetabling · Healthcare · Integrated optimization · Benchmarking

1 Introduction

Hospital resource optimization has progressed from solving isolated problems towards addressing integrated models that capture interdependencies between different planning and scheduling procedures. While prior research has addressed partial integrations, such as patient-to-room and nurse-to-patient assignment [1], fully integrated approaches remain scarce due to their computational complexity [3].

To stimulate research on these challenging problems, EURO working groups ORAHS[5] and PATAT[6] joined forces and organized the Integrated Healthcare Timetabling Competition 2024 (IHTC-2024). The competition introduced the

[5] Operational Research Applied to Health Services
[6] Practice and Theory of Automated Timetabling

Integrated Healthcare Timetabling Problem (IHTP), which combines Patient Admission Scheduling, Nurse-to-Room Assignment (NRA) and Surgical Case Planning (SCP) over a multi-week horizon with three daily shifts (early, late, night). The key decisions in the IHTP are i) selecting an admission day or postponing admission, for all patients ii) assigning patients to rooms iii) allocating nurses to rooms per shift and iv) assigning an operating theater for each patient's surgery. Various hard and soft constraints are considered, some of which relate to specific subproblems, while others arise from interactions among these subproblems. The deep integration of the three subproblems exposes the interdependencies between them. For example, patient admissions affect room and operating theatre occupancy, which in turn shape nurse and surgeon workloads and ultimately patient care.

The competition was announced at the ECCO 2024, ORAHS 2024 and PATAT 2024 conferences, and officially began on September 1st, 2024. The winners were announced in April 2025 and the award ceremony took place at the EURO 2025 conference in Leeds, UK (June 22–25, 2025). A detailed description of the problem, the competition setting and results is provided by Ceschia et al. [2], while all up-to-date information and data about the competition are available at `https://ihtc2024.github.io`.

2 Competition Design

The IHTC-2024 adopted a two-stage adjudication process to ensure fairness and robustness. After the first stage, five finalists were selected based on the mean rank of their submitted solutions for the *public* dataset. In the second stage, the organizers ran the finalists' algorithms on a *hidden* dataset under identical computational conditions (same hardware and time limits), preventing overfitting and eliminating hardware-related bias. By contrast to previous timetabling competitions, the IHTC-2024 allowed multi-core computation (up to four threads) and permitted the use of commercial solvers with academic licenses, better reflecting realistic optimization environments. Many participants embraced these new opportunities.

An important contribution of the competition is the development of a parametric instance generator capable of producing realistic and diverse scenarios, using 11 distinct parameters. A generation procedure based on the Hammersley point set ensures uniform coverage of the parameter space. The *public* dataset that was released at the start of the competition contains 30 instances with 30 to 500 patients each with a horizon of 2, 3 or 4 weeks. The *hidden* dataset consisted of an additional 30 instances of comparable dimensions, including previously unseen configurations to ensure an unbiased evaluation. Additionally, a validator tool was provided to verify feasibility and compute objective values, ensuring consistency across submissions.

Table 1. Finalists and optimization approaches.

Rank	Team	Method
1	v777v	Multi-stage MILP decomposition
2	SDU-IMADA	Metaheuristics
3	Twente	Hybrid MILP/CP/metaheuristics
4	ORTEC	Heuristic, MILP
5	UGent Mippets	ILP-based Local search, heuristic

MILP: Mixed-Integer Linear Programming; CP: Constraint Programming; ILP: Integer Linear Programming

3 Competition results

The competition attracted 32 teams, representing 16 countries across the Americas, Asia and Europe. Three main methodological categories emerged:

- **Mathematical Programming** combined with decomposition techniques;
- **Constraint Programming** integrated with heuristics;
- **Metaheuristics** primarily based on local search.

A clear trend toward hybridization was observed, with several teams combining multiple algorithmic paradigms.

Table 1 shows the final ranking of the five finalists based on their performance for the *hidden* dataset. SDU-IMADA achieved the best performance for the *public* instances, while v777v won the final evaluation for the *hidden* dataset. To quantify the disparity among finalists, a linear model was fitted to the ranks obtained for the hidden dataset. This statistical analysis confirms the superiority of the top two methods over others, with no significant difference between them, while also revealing strong instance-dependent performance variations. Importantly, no solutions were proven optimal, leaving room for additional research focusing on the computational challenge posed by the datasets.

4 Conclusions

The IHTC-2024 established a new benchmark for integrated healthcare timetabling, providing a solid foundation for future research on such complex problems. The wide variety of approaches and the lack of a universally dominant method both highlight the computational challenges posed by the problem. As future work, the benchmark could be complemented with real-world hospital instances to validate the generator's structural realism and further strengthen its value for future research.

References

1. Brandt, T., Klein, T.L., Reuter-Oppermann, M., Schäfer, F., Thielen, C., van de Vrugt, M., Viana, J.: Integrated patient-to-room and nurse-to-patient assignment in hospital wards. OR Spectrum (2025). https://doi.org/10.1007/s00291-024-00800-z
2. Ceschia, S., Rosati, R.M., Schaerf, A., Smet, P., Vanden Berghe, G., Zanazzo, E.: The Integrated Healthcare Timetabling Competition 2024. Operations Research, Data Analytics and Logistics **45**, 200481 (2025)
3. Rachuba, S., Reuter-Oppermann, M., Thielen, C.: Integrated planning in hospitals: a review. OR Spectrum **n.a.**, 1–54 (2024). https://doi.org/10.1007/s00291-024-00797-5

An Iterated Local Search Approach for One-Sided Crossing Minimization

Labeat Arbneshi[1] and Kadri Sylejmani[1⋆]

Faculty of Electrical and Computer Engineering,
University of Prishtina, Prishtina, Kosovo
{labeat.arbneshi, kadri.sylejmani}@uni-pr.edu

Abstract. The One-Sided Crossing Minimization (OSCM) problem seeks a vertex ordering of the free layer in a bipartite graph that minimises edge crossings—a core subroutine in the Sugiyama framework for layered graph drawing. We propose an Iterated Local Search (ILS) metaheuristic that combines six lightweight, *atomic* local search operators within an Adaptive Large Neighborhood Search (ALNS) selection scheme. Unlike conventional steepest-descent moves requiring quadratic time per invocation, each atomic operator evaluates a single candidate move in constant or near-linear time, enabling several thousand improvement steps per second. Experimental evaluation on 30 instances from the PACE 2024 Challenge demonstrates that our solver matches the best known results on 16 of 30 instances within 60 seconds—one-fifth of the competition budget—achieving a mean gap of 0.24 % across all thirty instances.

Keywords: Crossing minimization · Iterated local search · Graph drawing · ALNS · PACE 2024

1 Introduction

The OSCM problem is a fundamental task in layered graph drawing [1, 13]. Given a bipartite graph $G = (A \cup B, E)$ where layer A has a fixed order, the goal is to find a permutation π of layer B minimising edge crossings. Two edges (a_1, b_1) and (a_2, b_2) cross iff $\big(\mathrm{pos}_A(a_1) - \mathrm{pos}_A(a_2)\big) \cdot \big(\pi(b_1) - \pi(b_2)\big) < 0$. OSCM is NP-hard [2]; exact and parameterised algorithms [11, 12] exist but do not scale to the largest practical instances. The problem was the subject of the PACE 2024 Challenge [3, 14], which provided 100 instances of varying size (up to 91 000 edges) and structure, with a five-minute budget for the heuristic track.

Classical heuristics—*barycenter* and *median* [4]—sort B-vertices by the mean or median A-neighbour position. Local search refinements (adjacent-swap, insertion) close much of the gap but require $O(n^2)$ per pass. Our contribution is an ILS that replaces these steepest-descent operators with *atomic* (first-improvement, single-move) variants running in $O(n)$ or less, combined under an ALNS [5] weighting scheme.

⋆ Corresponding author.

Proceedings of the 15th International Conference on the Practice and Theory of Automated Timetabling (PATAT), Nottingham, UK, 25–28 August 2026

2 Approach

2.1 Initial Solution and ILS Framework

The initial ordering is obtained by computing both the barycenter and median heuristics and retaining the permutation with fewer crossings. The entire time budget is then allocated to ILS. The search proceeds in *phases* of random duration drawn from $\{1, 2, 3, 5, 8\}$ s. Within each phase, an operator is chosen by roulette-wheel sampling over the ALNS weights, applied to the current ordering, and accepted if the crossing count decreases. At the end of each phase the incumbent solution is either updated or perturbed. After ten consecutive phases without improvement the search restarts from the global best [6].

2.2 Atomic Operators

All six operators share the same contract: attempt *one* move and return the (ordering, cost) pair. Below, $\bar{d}$ denotes the average B-degree.
Adjacent Swap. Scan from a random position, accept the first swap with negative pairwise delta. $O(n)$ worst case.
Vertex Insert. Pick a random vertex u, sweep all insertion positions incrementally. $O(n \cdot \bar{d})$.
Non-Adjacent Swap. Pick random position i, scan $j \in [i{+}1, i{+}w]$ (w=10), accept the first improving swap. $O(w^2 \cdot \bar{d})$.
Segment Reversal. Pick random start, extend incrementally, accept first improving reversal of length $\ell \in [2, L_{\max}]$. $O(L_{\max}^2 \cdot \bar{d})$.
Or-opt-k (k=2, 3). Remove a random block of k vertices, sweep insertion positions. $O(n \cdot k \cdot \bar{d})$.

2.3 ALNS Weights and Perturbation

Operator weights are initialised with a bias toward insert and non-adjacent swap. After each phase, successful operators receive reward +3.0 per improvement; all weights decay by 0.85 (floor 0.1).

Perturbation uses two mechanisms: a *random kick* (mix of swaps, reversals, insertions, and block moves) and a *double-bridge* move [7] that reconnects four ordering segments as $A{+}C{+}B{+}D$. The double-bridge probability increases from 50% to 90% as phases pass without improvement; when deeply stuck ($\geq$8 phases), two double-bridges are applied in sequence.

3 Computational Results

We selected 30 instances from the PACE 2024 heuristic track covering six structural categories. All experiments used a 60-second limit on a single-threaded consumer machine; each instance was run up to three times (best retained). Table 1 compares against three top heuristic-track solvers on the `optil.io` leaderboard: CIMAT_Team (1st, memetic) [8], LUNCH (2nd) [9], and Arcee (4th) [10].

Table 1. Results on 30 PACE 2024 instances (60 s budget, best of 3 runs). Superscripts on Best Known: * = all three teams, C = CIMAT, A = Arcee, L = LUNCH.

Grp	Inst.	Best Known	CIMAT	Arcee	LUNCH	Ours	Δ(%)
Tiny	65	72 910 *	72 910	72 910	72 910	72 910	–
	49	779 *	779	779	779	779	–
	45	1 019 861 *	1 019 861	1 019 861	1 019 861	1 019 877	<0.01
	92	59 032 *	59 032	59 032	59 032	59 038	0.01
	66	103 362 *	103 362	103 362	103 362	103 373	0.01
Dense	14	1 442 485 *	1 442 485	1 442 485	1 442 485	1 442 485	–
	15	10 852 981 *	10 852 981	10 852 981	10 852 981	10 852 981	–
	37	33 588 819 C	33 588 819	33 588 877	33 588 850	33 597 500	0.03
	38	30 602 519 C	30 602 519	30 602 616	30 602 551	30 623 828	0.07
	86	1 117 722 *	1 117 722	1 117 722	1 117 722	1 117 967	0.02
Regular	33	440 799 *	440 799	440 799	440 799	440 799	–
	42	447 480 *	447 480	447 480	447 480	447 480	–
	43	772 760 *	772 760	772 760	772 760	772 760	–
	41	1 333 254 *	1 333 254	1 333 254	1 333 254	1 333 254	–
	80	159 436 *	159 436	159 436	159 436	159 458	0.01
Hub	57	252 004 *	252 004	252 004	252 004	252 004	–
	5	7 536 *	7 536	7 536	7 536	7 536	–
	7	10 837 *	10 837	10 837	10 837	10 837	–
	1	12 432 *	12 432	12 432	12 432	12 432	–
	9	177 606 *	177 606	177 606	177 606	177 606	–
Medium	11	2 030 125 *	2 030 125	2 030 125	2 030 125	2 031 398	0.06
	89	365 618 *	365 618	365 618	365 618	365 885	0.07
	19	11 031 578 C	11 031 578	11 031 597	11 031 592	11 050 038	0.17
	25	265 109 *	265 109	265 109	265 109	268 760	1.38
	39	748 666 C	748 666	748 673	748 667	769 682	2.81
Unbal.	34	0 *	0	0	0	0	–
	35	0 *	0	0	0	0	–
	71	562 159 *	562 159	562 159	562 159	568 451	1.12
	72	829 116 A,L	829 123	829 116	829 116	840 075	1.32
	29	0 *	0	0	0	0	–
Matched						**16/30**	
Avg. gap							**0.24%**

Our solver matches the best known result on 16/30 instances within 60 s (one-fifth of the competition budget). Results are strongest on hub (5/5) and regular (4/5) instances. The largest gaps appear on Medium-group instances (up to 2.81%), which benefit from longer search time. The atomic-operator design enables scalability: instance 9.gr ($|B|$=45k) runs in 60 s, whereas steepest-descent warm-up alone took over four minutes.

4 Conclusions and Future Work

We have presented an ILS solver for the One-Sided Crossing Minimization problem whose *atomic* operators enable thousands of moves per second, achieving competitive results against the top PACE 2024 participants within one-fifth of the competition time budget. Future work includes: (1) a hybrid strategy combining steepest descent for small instances with atomic operators for large ones;

(2) population-based extensions with crossover operators [8]; and (3) evaluation on the full 100-instance set.

References

1. Sugiyama, K., Tagawa, S., Toda, M.: Methods for visual understanding of hierarchical system structures. IEEE Trans. Syst. Man Cybern. **11**(2), 109–125 (1981)
2. Garey, M.R., Johnson, D.S.: Crossing number is NP-complete. SIAM J. Algebr. Discret. Methods **4**(3), 312–316 (1983)
3. Kindermann, P., Klute, F., Terziadis, S.: The PACE 2024 Parameterized Algorithms and Computational Experiments Challenge: One-Sided Crossing Minimization. In: Bonnet, É., Rzążewski, P. (eds.) IPEC 2024. LIPIcs, vol. 321, pp. 26:1–26:20. Schloss Dagstuhl (2024)
4. Eades, P., Wormald, N.C.: Edge crossings in drawings of bipartite graphs. Algorithmica **11**(4), 379–403 (1994)
5. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. **40**(4), 455–472 (2006)
6. Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search: Framework and applications. In: Handbook of Metaheuristics, 2nd edn., pp. 363–397. Springer (2010)
7. Martin, O., Otto, S.W., Felten, E.W.: Large-step Markov chains for the traveling salesman problem. Complex Syst. **5**(3), 299–326 (1991)
8. Segura, C., Lugo, L., Miranda, G., Serrano Cárdenas, E.D.: PACE Solver Description: CIMAT_Team. In: IPEC 2024. LIPIcs, vol. 321, pp. 31:1–31:4. Schloss Dagstuhl (2024)
9. Langedal, K., Bentert, M., Blanco, T., Drange, P.G.: PACE Solver Description: LUNCH – Linear Uncrossing Heuristics. In: IPEC 2024. LIPIcs, vol. 321, pp. 34:1–34:4. Schloss Dagstuhl (2024)
10. Boehmer, K., George, L.L., Hauser, F., Palarus, J.: PACE Solver Description: Arcee. In: IPEC 2024. LIPIcs, vol. 321, pp. 33:1–33:4. Schloss Dagstuhl (2024)
11. Nagamochi, H.: On the one-sided crossing minimization in a bipartite graph with large degrees. Theor. Comput. Sci. **332**(1–3), 417–446 (2005)
12. Dujmović, V., Fernau, H., Kaufmann, M.: Fixed parameter algorithms for one-sided crossing minimization revisited. J. Discret. Algorithms **6**(2), 313–323 (2008)
13. Tamassia, R. (ed.): Handbook of Graph Drawing and Visualization. CRC Press (2013)
14. PACE Challenge: One-Sided Crossing Minimization (2024). `https://pacechallenge.org/2024/`

Keyword Index

academic timetabling, 201
adaptive large neighbourhood search, 2, 445
agentic AI, 105
AI, 149
airline industry, 293
airline planning, 4
algorithm selection, 187
answer set programming, 121
assignment, 173
automated heuristic discovery, 5
automated timetabling, 297
avionics, 351

beam search, 425
benchmark, 93
benchmarking, 441
branch-price-and-cut, 321

candidate merge points, 235
capacity building, 201
case study, 311
chatbot, 149
code generation, 105
column generation, 343
constraint programming, 31, 105, 149, 231, 337, 381
convex hull, 173
counterfactual explanations, 381
course planning, 421
course timetabling, 3, 343, 345, 371
crew rostering, 411
crossing minimisation, 445

data-driven, 235
decision making, 249
decision support, 137
decomposition approach, 55
decomposition techniques, 293
delivery planning, 403
demand forecasting, 301
disruptiveness weighting, 45
dynamic aircraft arrival routes, 235
dynamic arc costs, 19
dynamic programming, 81, 161, 425
dynamic scheduling, 45, 219

educational timetabling, 187, 301
eHealth, 231
electric vehicle charging, 137
employee scheduling, 311
exact optimisation, 7
exam invigilation, 275
examination timetabling, 329
explainable AI, 329
explore/exploit balance, 67

fairness, 385, 391, 399
first-break-then-schedule, 321
fixed home dates, 385
flexible job shop scheduling, 375
formal methods, 7

genetic algorithm, 81, 173
genetic programming, 219
graph drawing, 445
great deluge, 355

healthcare, 289, 391, 441
heterogeneous workforce, 325
high school timetabling, 187
home healthcare routing and scheduling problem, 315
human-centred, 411
human-centric optimisation, 45
hyper-heuristics, 3, 329, 355, 403

iAdvisor, 149
incomplete round-robin scheduling, 395
infeasibility explanations, 381
infection modelling, 417
instance space analysis, 187
instructor scheduling, 345

integer programming, 235, 385
integrated optimisation, 441
interaction-oriented objectives, 275
International Timetabling Competition 2019, 67
iterated local search, 19, 445

knowledge distillation, 329

large language models, 5, 105, 329
large neighbourhood search, 337, 343
learning, 231
lexicographic optimisation, 137
logic-based Benders decomposition, 351
Luby's restart strategy, 67

maintenance constraints, 81
many-objective scheduling, 137
mathematical optimisation, 417
mathematical programming, 411, 435
metaheuristic optimisation, 375
metaheuristics, 3, 19, 307, 355, 425
minimal perturbation problem, 45
minimum-cost flow, 399
MiniZinc, 149
mixed-integer linear programming, 55
mixed-integer programming, 343
model reduction, 31

national coordination framework, 201
neighbourhood relation, 371
neighbourhoods, 307
nurse rostering, 161, 297, 391, 417
nurse scheduling problem, 231, 399

objective ordering, 137
omnichannel last-mile distribution, 365
operations research, 67
opponent assignment, 385
optimisation, 149, 231
OR in sports, 321, 395
overbooking, 301

PACE 2024, 445
parallel machine scheduling, 337
permutation flow shop, 263
permutation types, 263
personnel rostering, 55
personnel scheduling, 325
phenotypic characterisation, 219
policy evaluation, 249
practice, 407
preference elicitation, 411
prize-collecting vehicle routing problem, 2
production scheduling, 337
pruning rules, 7
pruning strategies, 161

Q-learning, 315

radiotherapy, 93
reentrant flow shop, 263
reinforcement learning, 5, 365, 403
resiliency, 4
rich scheduling, 337
rolling-window fairness, 385
rostering, 325
routing, 173
runway sequencing, 7

sand cat swarm algorithm, 375
satisfiability modulo theories, 7
scheduling, 93, 105, 149, 289, 293, 351, 407
SEIR, 417
self-directed learning, 421
self-rostering, 399
shift blocks, 399
shift coverage, 391
shift design, 325
shift transition, 417
simplicity, 3
simulated annealing, 263, 311, 325, 355, 365, 371, 425
sports scheduling/timetabling, 395
sports timetabling, 307, 321, 385

static Swiss tournament, 395
stochastic programming, 2
strategic planning, 435
student sectioning, 343, 421
study plan, 121
study regulations, 121
surrogate models, 81, 219
swarm intelligence, 375
symmetry handling, 321

team orienteering problem, 2
test laboratory scheduling problem, 381
time-sensitive networking, 351
timetabling, 149, 249, 275, 355, 407, 441
timetabling in home health care, 173
topological data analysis, 329
total completion time, 263
train dispatching, 31
transformers, 231
transport, 435
traveling tournament problem, 395
travelling salesman problem, 19
TV scheduling, 425

uncertainty, 4, 289
UniTime, 345
university course timetabling, 67
university timetabling, 45, 345
unrelated parallel machine scheduling, 81
upper secondary school timetabling, 421

vehicle routing, 3
vehicle routing problem, 365
verification, 7
virus optimisation algorithm, 315

warehouse planning, 3

XESTT, 297

Author Index

Abdennour Azerine, 81
Admir Kadriu, 19
Ahmed Kheiri, 249
Aida Aliu, 381
Aldy Gunawan, 365
Alexander Heide Madsen, 435
Alfred Nchimbi, 201
Ali Hassanzadeh, 385
Andrea Schaerf, 371, 441
Andreas Krystallidis, 187
Angela Hsiang-Ling Chen, 315
Azuraliza Abu Bakar, 375

Bence Kovari, 31
Boubaker Daachi, 231
Bruno Chinaglia, 55

Chenwei Jin, 219
Chiara Camilla Rambaldi Migliore, 93
Ching-Hsuan Lin, 365
Chiu-Yuan Chen, 173
Christiane Schmidt, 235

Daniel Karapetyan, 403
Daniel Torres Ruiz, 403
David Kirpekar-Sauer, 289
David Pisinger, 2
David Van Bulck, 307, 321, 395
Dima Nazzal, 301
Douglas Nascimento, 55
Dries Goossens, 307, 391, 395

Egon L. van den Broek, 411
Elad Shufan, 263
Elina Rönnberg, 235, 351
Eljon Shala, 425
Ender Ozcan, 329, 403
Ender Özcan, 355
Endrita Vllasaliu, 425
Erwin Abbink, 411
Eugenia Zanazzo, 441

Fatima Iken, 81
Felix Winter, 337
Fisnik Hazrolli, 425
Florian Mischek, 381
Florian Strohm, 105
Francesca Da Ros, 371
Franklina Toledo, 55

Geert De Maere, 7
Gerhard Post, 407
Giovanni Iacca, 93
Greet Vanden Berghe, 55, 161, 391, 441
Günther Raidl, 137

Hagai Ilani, 263
Hana Rudová, 3
Hannes Uppman, 351
Helen Kirby-Hawkins, 45
Hendrik Bierlee, 293
Henry Otunuya, 121
Hicham Kamhi, 311

Ida Gjergji, 293
Itamar Segal, 263

Jannik Schwab, 105
Jason Atkin, 7
Jasper van Doornmalen, 321
Jeffrey Kingston, 297
Jeremie Clos, 329
Jeroen Mulder, 4
Jesper Vines, 351
Joel Sokol, 301
Jourdan D'Orville, 67

Kadri Sylejmani, 19, 425, 445
Kaltrina Krasniqi, 425
Karel Devriesere, 307, 395
Kevin Tierney, 5
Konstantinos G. Zografos, 249
Kuan-Fu Chen, 365

Labeat Arbneshi, 425, 445

Larbi Boubchir, 231
Laurent Drapeau, 149
Laurenz Tomandl, 137
Leshan Tan, 219
Lhassane Idoumghar, 81
Lisa Garcia Tercero, 391
Lotte Berghman, 311
Luca Di Gaspero, 371
Lucas Kletzander, 293, 325
Lukas Frühwirth, 337
Lukas Wolf, 343
László Kálmán Trautsch, 31

Mahmoud Golabi, 81
Maksym Moroz, 235
Marco Huber, 105
Marco Roveri, 93
Marjan van den Akker, 411
Matthew Davison, 249
Melhem El Helou, 149
Michael Hölscher, 421
Mikaela Lindberg, 351
Mingxuan Li, 67
Mio Hourai, 417
Mohamed A. Madani, 231
Mohd Nor Akmal Khalid, 375
Mutsunori Yagiura, 417

Neel Gaglani, 355
Neema Kumburu, 201
Nilgun Sengoz, 329
Nysret Musliu, 93, 187, 293, 337, 381

Olivia Phillips, 301

Patrick Wagner, 105
Peter J. Stuckey, 293
Pieter Smet, 161, 289, 391, 441

Qing Chuan Ye (registered as Charlie Ye), 399

Razali Yaakob, 67
Reinout Annaert, 275
Renan S. Maglasang, 365
Roberto Maria Rosati, 441
Robin Tourlamain, 161
Roghayeh Hajizadeh, 235
Rong Qu, 219, 403
Rubén Ruiz-Torrubiano, 187
Ruibin Bai, 219

Salwani Abdullah, 67
Sara Ceschia, 371, 441
Say Leng Goh, 67
Sebastian Gunawan, 315
Sebastian Schellhorn, 121
Shefket Bylygbashi, 425
Shih-Wei Lin, 365
Shihhung Su, 173
Sina Abdipoor, 67
Steffen Limmer, 137

Tal Grinshpoun, 263
Tatiana Polishchuk, 235
Thomas Stidsen, 435
Thomas Weise, 67
Tobias Rodemann, 137
Toby Clark, 7
Tomas Muller, 345
Tony Rouhayem, 149

Uran Lajqi, 425

Vanny Minanda, 315
Veerle van den Hurk, 411
Victor Alexander Brandsen, 435
Victoria Kessy, 201
Vincent F. Yu, 365

Walid Abdelaidoum, 231
Wei Wu, 417
Wirawati Dewi Ahmad, 375

Xinan Chen, 219

Yll Berisha, 425
Youssef Bakouny, 149
Yun-Chia Liang, 315