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.

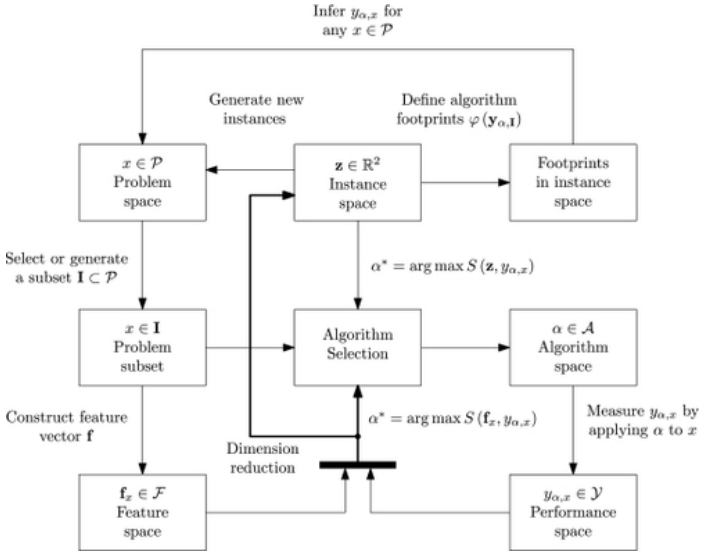


Fig. 1: Instance Space Analysis Framework by Smith-Miles et al. [8] extending the Algorithm Selection Framework by Rice [7]

.

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-

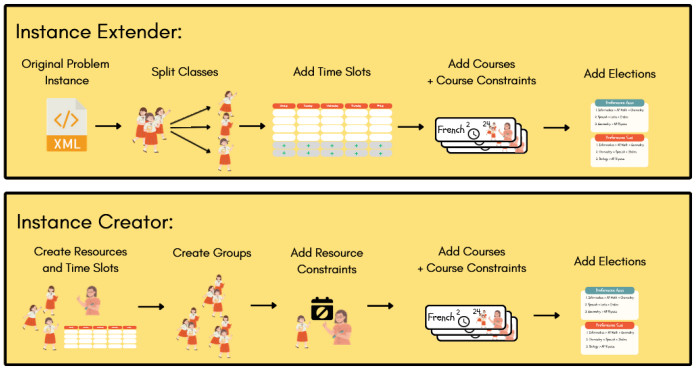


Fig. 2: Instance Generator and Instance Extender pipeline.

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.

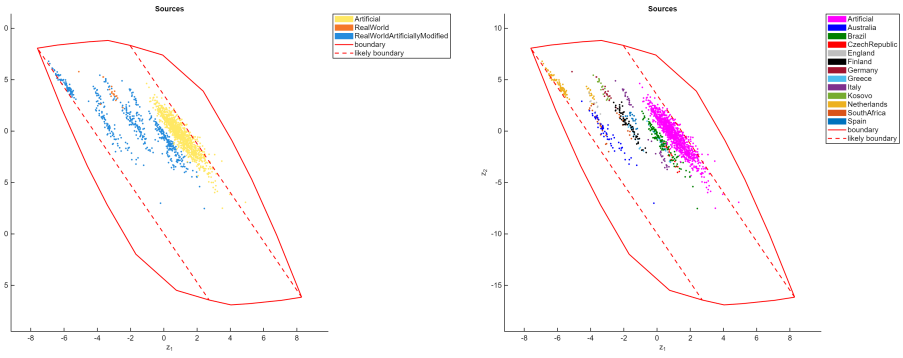
Overall, the resulting feature space jointly captures instance scale, elective modularity, resource contention, constraint tightness, and interaction topology, providing a comprehensive numerical characterization suitable for ISA.

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

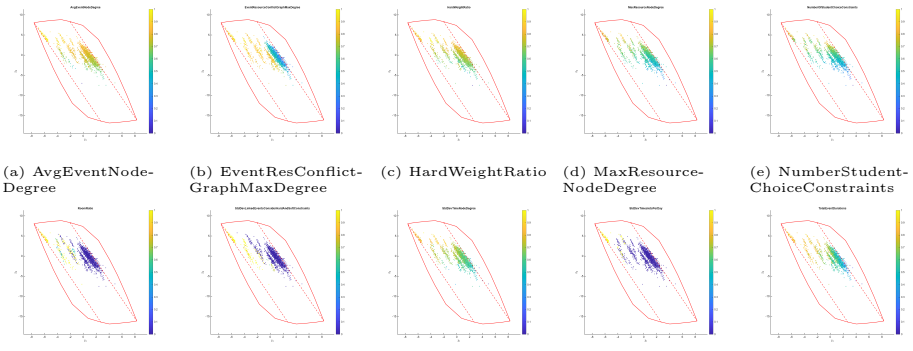


(a) Colored based on method of generation.

(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).

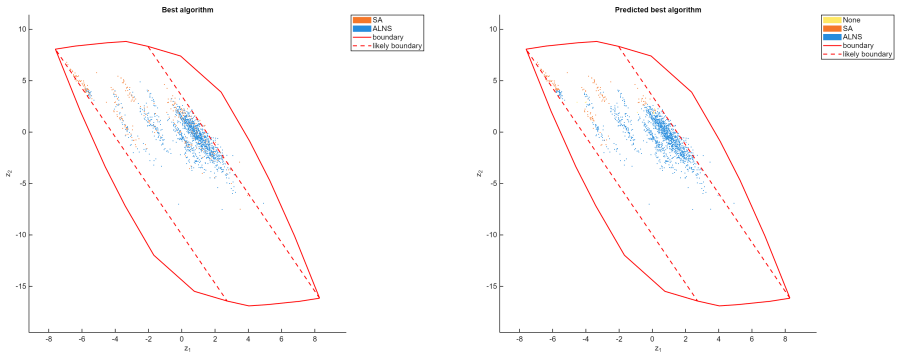
$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} -0.3938 & -0.043 \\ -0.3421 & 0.1236 \\ -0.3029 & 0.1451 \\ -0.2087 & 0.5812 \\ -0.41 & -0.1036 \\ -0.126 & 0.5067 \\ -0.1294 & 0.4098 \\ -0.2404 & 0.3463 \\ -0.1212 & 0.4381 \\ -0.4328 & -0.0724 \end{bmatrix}^T \cdot \begin{bmatrix} RoomRatio \\ StdDevTimeslotsPerDay \\ TotalEventDurations \\ NumberOfStudentChoiceConstraints \\ StdDevLinkedEventsAmount \\ HardWeightRatio \\ MaxResourceNodeDegree \\ StdDevTimeNodeDegree \\ AvgEventNodeDegree \\ EventResConflictGraphMaxDegree \end{bmatrix} \quad (1)$$

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