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.

Algorithm 1

```
1: procedure RS(f : S ↦ ℕ0)
2:     randomly sample TTb from S;  yb ← f(TTb)
3:     while computational budget not exhausted do
4:         randomly sample TTn from S;  yn ← f(TTn)
5:         if yn ≤ yb then
6:             TTb ← TTn;  yb ← yn        ▷ preserve best solution
7:     return TTb, yb
```

Algorithm 2

```
1: procedure HC(f : S ↦ ℕ0)
2:     randomly sample TTb from S;  yb ← f(TTb)
3:     while computational budget not exhausted do
4:         TTn ← copy of TTb with 1 class randomly changed
5:         yn ← f(TTn)
6:         if yn ≤ yb then TTb ← TTn;  yb ← yn
7:     return TTb, yb
```

Algorithm 3

```
1: procedure HRSHCL(f : S ↦ ℕ0, ξ ∈ ℕ1)
2:     yb ← ∞;  i ← 1;  j ← 1                                   ▷ luby(1) = 1
3:     while computational budget not exhausted do
4:         j ← j − 1
5:         if j = 0 then                    ▷ perform restart (always true in first iteration)
6:             randomly sample TTc from S;  yc ← f(TTc)
7:             i ← i + 1;  j ← ξ ∗ luby(i)
8:         else
9:             TTn ← copy of TTc with one class randomly changed
10:            yn ← f(TTn)
11:            if yn ≤ yc then TTc ← TTn;  yc ← yn
12:        if yc ≤ yb then TTb ← TTc;  yb ← yc          ▷ preserve best-ever solution
13:    return TTb, yb
```

$$\text{where luby}(i) = \begin{cases} 2^{k-1} \text{ if } i = 2^k - 1 \\ \text{luby}(i - 2^{k-1} + 1) \text{ if } 2^{k-1} \leq i < 2^k - 1 \end{cases} \quad (4)$$

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

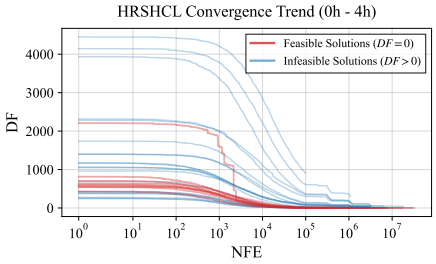

Fig. 2. Distance to Feasibility vs. NFE for HRSHCL (Initial 4 Hours)

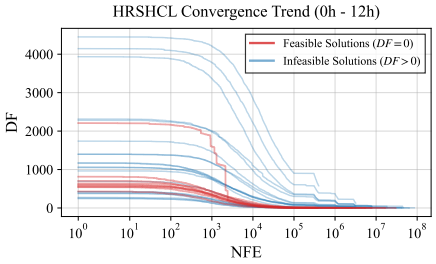

Fig. 3. Distance to Feasibility vs. NFE for HRSHCL (Full 12 Hours)

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)