Indonesian Journal of Electrical Engineering and Informatics (IJEEI)
Vol. 8, No. 1, March 2020, pp. 163~177
ISSN: 2089-3272, DOI: 10.11591/ijeei.v8i1.1660

Performance Analysis of Graph Heuristics and Selected Trajectory Metaheuristics on Examination Timetable Problem

Ashis Kumar Mandal[1], M.N.M. Kahar [2]
[1]Graduate School of Software and Information Science, Iwate Prefectural University, Iwate, Japan
[2]Faculty of Computer Systems & Software Engineering, Universiti Malaysia Pahang, Pahang, Malaysia

Article Info

Article history:

Received Sep 24, 2019
Revised Feb 4, 2020
Accepted Feb 14, 2020

Keywords:

Capacitated,
Combinatorial optimization,
Exam timetabling problems,
Graph heuristic,
Trajectory meta-heuristics,
Un-capacitated

ABSTRACT

Examination timetabling problem is hard to solve due to its NP-hard nature, with a large number of constraints having to be accommodated. To deal with the problem effectually, frequently heuristics are used for constructing a feasible examination timetable, while meta-heuristics are applied for improving the solution quality. In this paper, we present the combination of graph heuristics and major trajectory metaheuristics for addressing both capacitated and un-capacitated examination timetabling problem. For constructing the feasible solution, six graph heuristics are used. They are largest degree (LD), largest weighted degree (LWD), largest enrolment degree (LE), and three hybrid heuristics with saturation degree (SD) such as SD-LD, SD-LE, and SD-LWD. Five trajectory algorithms comprising of tabu search (TS), simulated annealing (SA), late acceptance hill-climbing (LAHC), great deluge algorithm (GDA), and variable neighborhood search (VNS) are employed for improving the solution quality. Experiments have been tested on several instances of un-capacitated and capacitated benchmark datasets, which are Toronto and ITC2007 datasets, respectively. Experimental results indicate that hybrid graph heuristics produce the best initial solutions across both these benchmark problems. The study also reveals that, during improvement, GDA, SA, and LAHC can produce better quality solutions compared to TS and VNS for solving both benchmark datasets. Moreover, the performance of our method is comparable with other approaches in the scientific literature.

Corresponding Author:

Ashis Kumar Mandal
Graduate School of Software and Information Science,
Iwate Prefectural University
152-52 Sugo, Iwate, Japan.
Email: g236r004@s.iwate-pu.ac.jp

1. INTRODUCTION

In the last few decades, the examination timetabling problem has been studied vastly in the artificial intelligence (AI) and operational research (OR) communities due to its complexity and practical significance in educational institutions [1]. Academic institutions frequently face a considerable amount of challenges in the effective scheduling of their examinations with limited resources in a reasonable time. An examination timetabling is a system of allocating a set of examinations into a limited number of time slots and rooms in order to satisfy all hard constraints and to minimize the soft constraint violations as much as possible. The satisfaction of all hard constraints leads to a feasible solution, while the quality of a solution depends on soft constraint satisfaction. It is observed that, according to the requirements and resources of educational institutions, these constraints can be different [2, 3].

Examination timetable problems can be categorized as capacitated and un-capacitated problems [4]. In an un-capacitated branch, room capacity is not considered. In a capacitated variant, however, room capacity is considered as a hard constraint. Usually, capacitated datasets are more complex to solve than un-capacitated datasets. Also, capacitated datasets closely resemble the real world timetabling problem, and they are highly constrained. An example of an un-capacitated problem is Toronto datasets, whereas ITC2007 datasets are a capacitated problem [5].

Educational institutions require considerable work to generate an examination timetable due to accommodating examinations, which frequently conflict with each other, into limited resources being inherent difficulties. Moreover, the number of different constraints and the size of the examination instances makes examination timetabling more complex to solve. Violating constraints, such as assigning two conflicting examinations at the same time, is fatal and directly affects students' careers. Considering all of the constraints, an optimal solution is rare to be obtained in a reasonable time, and therefore, researchers focus on sub-optimal solutions, i.e., quality solutions [6]. Examination timetabling is the ideal example of a combinatorial optimization problem where the best solution has to be searched from a very large solution space. Lately, considerable attention has been driven to heuristic and meta-heuristic search techniques for addressing the examination timetabling problem. This is because, for many larger instances (examinations) in real-world settings, these search techniques are capable of finding a good quality solution in reasonable time and limited resources. These include graph-based sequential techniques [7], trajectory-based meta-heuristics[8], population-based meta-heuristics [9], and recently hyper-heuristics [10]. Detail description of various methods related to examination timetabling can be found in different surveys in the examination timetabling domain [11] [12, 13] as well as in PATAT series of conference proceedings held from 1995 to 2018 (available at http://www.patatconference.org/).

This paper focuses on graph colouring heuristics and well-known trajectory metaheuristics for addressing the examination timetabling problem. In the first step, feasible solutions are constructed using six graph heuristic algorithms. The first three approaches are largest degree (LD), largest weighted degree (LWD), largest enrolment degree (LE). The rest of the three is the hybridization of above three with a saturation degree (SD) separately producing SD-LD, SD-LWD, and SD-LE. In the second step, the graph heuristic algorithm that produces the best quality solution is used for constructing an initial solution for each of the trajectory meta-heuristic algorithms. Each meta-heuristic then optimize the solution vector and produce quality solutions. Five trajectory algorithms, which are tabu search (TS), simulated annealing (SA), late acceptance hill-climbing (LAHC), great deluge algorithm (GDA), and variable neighborhood search (VNS), are tested on two popular and complex benchmark datasets, namely Toronto datasets and ITC2007 datasets.

The remaining of this paper is organized as follows. Section 2 describes the examination timetabling problem and mathematical formulation. Section 3 presents related works on the examination timetabling problem. Section 4 describes the heuristic and metaheuristic approaches being investigated. Section 5 contains materials and methods of the study, while section 6 presents experimental results. Finally, section 7 draws the overall conclusion.

2. EXAMINATION TIMETABLING PROBLEM AND FORMULATION

In this paper, two examination timetabling benchmark datasets are used to assess the performances of proposed approaches. The datasets are Toronto benchmark dataset, un-capacitated datasets, and ITC2007 benchmark datasets, capacitated datasets. The motivation for using these datasets is that they are highly studied in research communities, and ITC2007 is more realistic and more complex. The description of two examination timetabling benchmarks is given below.

2.1. Un-capacitated benchmark datasets

The most widely used examination benchmark datasets were introduced by Carter and Laporte [14]. These datasets are also known as Toronto datasets. It is available from http://www.asap.cs.nott.ac.uk/resources/data.shtml. These datasets are un-capacitated examination timetabling benchmark datasets, assuming an unlimited number of seats are available during exam assignments. The Toronto datasets consist of 13 problem instances. Table 1 summarises the datasets.

The Toronto examination timetable hard constraint insists that no students are allowed to attend two or more exams simultaneously (also known as the clashing constraint). The soft constraint (i.e., how the quality of the timetable is measured) is to spread the exams evenly for all students.

The objective function is shown in Eq.1. A penalty value of 16 is given for assigning two examinations consecutively for a given student. A penalty value of 8 is assigned if there is one timeslot between exams followed by a penalty value 4, 2 and 1 for 2, 3 and 4 timeslot gaps between exams, respectively.

$$\min \frac{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij} \times proximity\,(t_i,t_j)}{M} \quad (1)$$

Where

$$proximity(t_i,t_j) = \begin{cases} \dfrac{2^5}{2^{|t_i-t_j|}} & if\ 1 \le |t_i - t_j| \le 5 \\ 0 & otherwise \end{cases}$$

N is the number of examinations
M is the total number of students
T is the number of available timeslots.
c_{ij} is the conflict matrix, where each element in the matrix is the number of students taking examination i and j, and where $i, j \in \{1,..., N\}$ {Qu, 2009 #444}
t_k ($1 \leq t_k \leq T$) specifies the assigned timeslot for examination k ($k \in \{1,..., N\}$)

The objective for the Toronto datasets problem is to satisfy the hard constraint and minimize the penalty value of the soft constraint violations

Table 1. Toronto benchmark datasets

Instances	No. of timeslots	No. of exams	No. of students	Conflict density
car-s-91	35	682	16925	0.13
car-f-92	32	543	18419	0.14
ear-f-83	24	190	1125	0.27
hec-s-92	18	81	2823	0.42
kfu-s-93	20	461	5349	0.06
lse-f-91	18	381	2726	0.06
pur-s-93	42	2419	30029	0.03
rye-s-93	23	486	11483	0.07
sta-f-83	13	139	611	0.14
tre-s-92	23	261	4360	0.18
uta-s-92	35	622	21267	0.13
ute-s-92	10	184	2750	0.08
yor-f-83	21	181	941	0.29

2.2. Capacitated benchmark datasets

The 2nd international timetable competition (ITC2007) examination datasets were established to facilitate researchers to explore real-world examination timetabling problem and to reduce the gap between theory and practice. The ITC2007 examination datasets contain eight instances (see Table 2), comprising a variety of hard and soft constraints. Referring to Table 2, A1 is the number of students registered, A2 shows the number of exams, A3 is the number of timeslots, A4 indicates the number of available rooms, A5 is the period hard constraints, A6 is the room hard constraints, and A7 is the conflict density. They are available for download from the link http://www.cs.qub.ac.uk/itc2007/examtrack.

Table 2. ITC2007 Benchmark datasets

Instances	A1	A2	A3	A4	A5	A6	A7
Exam_1	7,833	607	54	7	12	0	5.05%
Exam_2	12,484	870	40	49	12	2	1.17%
Exam_3	16,365	934	36	48	170	15	2.62%
Exam_4	4,421	273	21	1	40	0	15.0%
Exam_5	8,719	1018	42	3	27	0	0.87%
Exam_6	7,909	242	16	8	23	0	6.16%
Exam_7	13,795	1096	80	15	28	0	1.93%
Exam_8	7,718	598	80	8	20	1	4.55%

The hard constraints for ITC2007 examination datasets are defined as follows:
H1. One student can sit only one exam at a time.
H2. The capacity of the exam will not exceed the capacity of the room.
H3. The exam duration will not violate the period duration.
H4. Three types of exam ordering must be respected.
- *Precedences:* exam *i* will be scheduled before exam j.
- *Exclusions:* exam *i* and exam *j* must not be scheduled at the same period.
- *Coincidences:* exam *i* and exam *j* must be scheduled in the same period.

H5. Room exclusiveness must be maintained. For example, exam *i* must take place only in room number 206.

Soft constraints for ITC2007 examination datasets are summarized as follows:
S1. *Two Exams in a Row (C_s^{2R}):* Restriction is imposed for a student to sit successive exams on the same day.
S2. *Two Exams in a Day(C_s^{2D}):* Restriction is imposed for a student to sit two exams in a day.

S3. *Spreading of Exams(C_s^{PS})*: Spread the exams as evenly as possible over the periods so that the preferences of students, such as avoiding closeness of exams are preserved.
S4. *Mixed Durations(C^{NMD}):* Avoid scheduling exams of different durations in the same room and period.
S5. *Scheduling of Larger Exams (C^{FL}):* Restriction is imposed to assign larger exams at the late of the timetable.
S6. *Room Penalty (C^R):* Some rooms are restricted for assigning some exams with the associated penalty.
S7. *Period Penalty(C^P): S*ome periods are restricted for assigning some exams with the associated penalty.

The objective of solving these instances is to satisfy the hard constraints and minimize the soft constraint violations (penalty) as much as possible in order to produce a good quality timetable. The objective function can be formularized as in Eq.2 [15].

$$min \sum_{s \in S}(W^{2R}C_s^{2R} + W^{2D}C_s^{2D} + W^{PS}C_s^{PS}) + W^{NMD}C^{NMD} + \ W^{FL}C^{FL} + C^P + C^R \quad (2)$$

Where W indicates weight for each soft constraint, and S defines the set of students. Table 3 shows the weights for the ITC2007 examination datasets. Note that the weights are not included in C^P and C^R in the equation. This is because these weights are already included in their definition. A more detailed description of this examination track, as well as their objective functions, can be found in [15, 16]

Table 3. Weights of the ITC2007 examination datasets

Instances	weight for two in a day (W^{2D})	weight for two in a row (W^{2R})	weight for period spread (W^{PS})	weight for no mixed duration (W^{NMD})	weight for the front load penalty (W^{FL})
Exam_1	5	7	5	10	5
Exam_2	5	15	1	25	5
Exam_3	10	15	4	20	10
Exam_4	5	9	2	10	5
Exam_5	15	40	5	0	10
Exam_6	5	20	20	25	15
Exam_7	5	25	10	15	10
Exam_8	0	150	15	25	5

3. RELATED WORKS

The examination timetabling problem has been widely investigated, and a wide range of approaches have been reported in AI or OR literature over the last few decades. Popular techniques used often for solving examination timetabling are described below.

The examination timetabling literature focuses on graph heuristics frequently because they are simple and tend to be useful in constructing a feasible solution quickly. Kahar and Kendall [3] used four graph heuristics, which are LD, LWD, LE, and SD, to solve University Malaysia Pahang examination timetabling problem. The authors reported that these heuristics produced feasible solutions for all instances of the datasets and better quality solutions than the university's existing software. Sabar et al. [17] used graph colouring hyper-heuristic for constructing an examination timetable. In their approach, four lists were prepared using hybridization of low-level heuristics, and 'difficulty index' (a parameter) was issued for the selection of examinations for scheduling. Abdul Rahman et al. [18] proposed the adaptive ordering strategy whereby adaptive mechanism was enabled by adding a heuristic modifier to graph heuristics. Another work is a fuzzy graph heuristic, where a fuzzy combination of LD, SD, and LE was investigated for ordering examinations [19]. Besides, several graph colouring approaches were hybridized with hill climbing for successfully solving both ITC 2007 and Toronto datasets [20, 21].

Pais and Amaral [22] implemented an improved tabu search for the examination timetabling. Here, tabu list is automatically tuned by a fuzzy inference rule-based system (FIRBS), and this improves tabu search in exploring a promising area of solution space. Abdullah et al. [23] hybridized tabu search with a memetic algorithm for solving university timetabling problem. This algorithm employed a set of neighbourhood structures that was controlled using a tabu list.

Simulated annealing has been extensively used for examination timetabling problem. Battistutta et al. [24] presented simulated annealing with a feature-based tuning approach for solving ITC2007 examination timetabling. The tuning stage was started by selecting the most important parameters. Then a regression model was developed that correlated the value of the most important parameter to the features of the instances. Results indicate that proper tuning can produce competitive results. Simulated annealing for solving examination timetabling can also be found in [25] and [26].

Burke and Bykov [27] observed that late acceptance hill-climbing produced better solutions than other local search methods while implemented in Toronto and ITC2007 datasets. Subsequently, Alzaqebah and Abdullah [28] combined late-acceptance hill-climbing as well as simulated annealing with bee colony optimization algorithms for solving Toronto and ITC2007 datasets. Besides, Bykov and Petrovic [29] recently have proposed another single-parameter local search similar to LAHC, which is step counting hill climbing. The approach was tested on all instances of ITC2007 examination datasets and produces some good results.

Another successful metaheuristic for the examination timetabling problem is great deluge algorithm. Kahar and Kendall [30] proposed a modified-great deluge for solving UMP examination timetabling problem. They used a dynamic decrease of boundary level, and when there was no improvement for certain iterations, the boundary level was increased. They conducted experiments with different initial solutions and neighbourhood structures. Mandal and Kahar [8] proposed a novel approach where partially selected examinations were constructed using hybrid graph heuristics, and then these scheduled examinations were improved using a modified great deluge algorithm. Great deluge algorithm was hybridized with an electromagnetic-like mechanism for moving simple solution(s) to high-quality solution(s) avoiding local optima [31].

Population-based metaheuristics work with more than one solution for the optimization process. Pillay and Banzhaf [32] implemented an informed genetic algorithm (IGA) to solve Toronto benchmark datasets. A two-phase approach was used. In the first phase, the timetable was evolved with satisfying hard constraints, and soft constraints were considered during the improvement phase. In both cases, genetic algorithm was employed to be guided by some domain knowledge. Results indicate that IGA tends to produce better examination timetabling than other evolutionary algorithms. Hosny and Al-Olayan [33] proposed a mutation-based genetic algorithm for examination timetabling problem whereby crossover was avoided, and mutation was used as the main genetic operator during the evolutionary process. Alinia Ahandani et al. [34] investigated the discrete PSO algorithm for solving the examination timetabling problem. The particles' positions were updated using genetic operations like mutation and crossover. The quality of particles' position was improved using three approaches of local search applied to hybridize discrete particle swarm optimization. The approach showed satisfactory results while tested on the Toronto datasets. Sainte and Larabi [35] proposed a hybrid PSO that produces stable solutions for examination timetabling. Alzaqebah and Abdullah [28] used the artificial bee colony algorithm by incorporating late acceptance hill-climbing, adaptive approach in neighbourhood selection, and disrupting selection strategy. They observed that disrupting selection strategy diversifies the population and prevents early convergence. Bolaji et al. [36] proposed a hybridization of an artificial bee colony with a local search and harmony search algorithm for solving un-capacitated examination timetabling.

Memetic algorithms are well-known population-based approaches that are known as the hybridization of evolutionary algorithms and local search methods. The combination of genetic algorithm with modified violation directed hierarchical hill climbing (VDHC) was used for solving examination timetabling problem [37]. Memetic algorithm for examination timetabling problem was also found in research conducted by Abdullah and Turabieh [38], Lei et al. [39], and Leite et al. [40].

Hyper-heuristic is a relatively new domain that can effectively solve the educational timetabling problem. It is the domain independence high-level search strategy that modifies solutions indirectly by adequately selecting and employing some low-level heuristics Pillay [41]. Anwar et al. [42] investigated the harmony search hyper-heuristic approach for solving the ITC2007 examination problem. A basic harmony search algorithm was employed in the high-level heuristic, which controlled the low-level heuristics. These were two neighbourhood structures: move and swap operation on examinations. Results revealed that the approach can produce some competitive results compared to state-of-the-art approaches. Demeester et al. [43] presented a hyper-heuristic framework based on the mechanism of tournament selection in genetic algorithms. In low-level heuristic, the number of moves operations based on problem types was selected. Recently, Muklason[10] has proposed hyper-heuristic for multi-objective examination timetabling problem.

4. GRAPH COLORING HEURISTICS AND TRAJECTORY META-HEURISTICS

In examination timetabling literature, AI techniques like heuristics and meta-heuristics are used for solving examination timetabling. For instance, heuristics are often employed for solution construction. These heuristics contain domain-specific knowledge that guides search in finding feasible timetabling or even better timetabling from the solution space. Meta-heuristics are usually used widely to optimize the timetabling. They are problem-independent algorithms that guide subordinate heuristics with some intelligent strategies for exploring and exploiting the search space so that efficient solutions (optimal or near-optimal) can be found [44]. It is often classified into two main branches, including trajectory-based (i.e., local search meta-heuristics) and population-based approaches. Trajectory-based methods take one single solution at a time and explore the search space to generate near-optimal solutions. Simulated annealing and tabu search are the two examples of

trajectory-based methods. In population-based search, more than one different initial solutions (i.e., population) are considered at a time for generating near-optimal solutions. Genetic algorithm and ant colony algorithm are two population-based approaches. The algorithms used in this study are described below.

4.1. Graph Heuristics (GH)

Examination timetabling problem can be modeled using a graph colouring algorithm, and therefore, it exhibits similarity with graph colouring problem [45]. In graph colouring problem, an undirected graph is a representation comprising a set of n vertices. In graph colouring problem, an undirected graph $G = (V, E)$ is a representation comprising a set of n vertices $V = (v_1, v_2, v_3, \dots, v_n)$ and a set of edges E. If (v_i, v_j) is an edge in a graph $G = (V, E)$ then vertex v_i is adjacent to vertex v_j. The graph colouring problem involves assigning k-colours in $V = (v_1, v_2, v_3, \dots, v_n)$ such that no two adjacent vertices are assigned the same colour. It is straightforward to convert the graph colouring problem to the examination timetable problem (vice-versa) by considering all vertices as events (i.e., examinations) and an edge between any pair of vertices as conflicting examinations. That is, these examinations have taken at least one student and do not exist in the same time slot. Finally, k-colours is equivalent to the number of time slots.

In graph theory, colouring a graph with a predefined limited number of colours (k-colouring of a graph) is complex tasks. This graph problem is in the class of NP-complexity [46, 47]. Graph colouring heuristics (Graph heuristics) are popular sequential approaches that are used for constructing an initial timetable [17]. As the brute-force approach of solving graph colouring problem is NP-hard, graph heuristics encompass some heuristic colouring techniques, such as vertex ordering, to find optimal or near-optimal colourings in polynomial time. In the context of examination timetabling, graph heuristics are based on ordering strategies where examination with most 'difficulty' is chosen for scheduling first so that finally, a feasible solution can be obtained. The examination difficulty is measured with various graph heuristic techniques. The most commonly used graph heuristics ordering strategies seen in the literature are described as follows:

- Largest degree (LD): In this ordering, the number of conflicts is counted for each examination by checking its conflict with all other examinations. Then, examinations are ordered in decreasing manure such that exams with the largest number of conflicts come fast.
- Largest weighted degree (LWD): This ordering has a similarity with LD. The difference is that in the ordering process, the number of students associated with the conflict is considered.
- Largest enrolment (LE): The examinations are ordered decreasingly with the value of registered students of these examinations.
- Saturation degree (SD): Examination ordering is based on the availability of remaining time slots where unscheduled examinations with the lowest number of available time slots for scheduling are given priority for scheduling first. The ordering is dynamic as it is updated after scheduling each exam.

4.2. Tabu search (TS)

Tabu search is a local search meta-heuristic algorithm, which is firstly proposed by Glover [48]. The basic mechanism of tabu search is based on hill-climbing algorithm. However, it can avoid trapping into local optima by accepting the worst solutions. A memory structure called tabu list is used for avoiding the exploration of the same neighbourhood solutions for a certain number of iterations. In other words, tabu list occupies recently visited solutions and remains it tabu for avoiding cycling. A mechanism like aspiration criteria is also used to allow promising solutions with tabu free status if the penalty value of the solution vector is better than that of the current best-known solution. Figure 1 illustrates the simple tabu search approach.

```
1.  Create initial solution s
2.  Initialize tabu list T
3.  while termination criterion not satisfied do
4.      Determine complete neighbourhood N of current solution s
5.      Choose best non-tabu solution s' from N
6.      Switch over to solution s' (current solution s is replaced by s')
7.      Update tabu list T
8.      Update best found solution (if necessary)
9.  end while
10. return Best found solution
```

Figure 1. Tabu search procedure.

4.3. Simulated annealing (SA)

Simulated annealing (SA) is a local search meta-heuristic technique based on a physical annealing process that probabilistically accepts some worst solutions to escape from the local optimum. It was introduced by Kirkpatrick and Vecchi [49]. Simulated annealing starts with a randomly generated initial solution, and in each iteration, it tries to improve the solution quality. If the neighbouring solution is better than or equal to the current solution, it is replaced with the current one. Otherwise, acceptance of neighbouring solution is decided on probability function $\exp(-(f(s')-f(s))/T)$, where $f(s')$ is neighbouring solution, $f(s)$ is current solution, and T is a parameter known as temperature. Initially, the algorithm starts with a high T and periodically decreases the value using the cooling schedule until the temperature is zero or any terminal condition. Figure 2 illustrates the simulated annealing process for the minimization problem.

```
1.  Choose, at random, an initial solution S for the system to be optimized
2.  Initialize the temperature T
3.  while the stopping criterion is not satisfied do
4.      repeat
5.          Randomly select  S' ∈ N(S)
6.          if f(S') ≤ f(S) then
7.              S ← S'
8.          else
9.              S ← S' with a probability exp(−(f(s')−f(S))/T)
10.         end if
11.     until the "thermodynamic equilibrium" of the system is reached
12.     Decrease T
13. end while
14. return  the best solution met
```

Figure 2. Simulated annealing procedure.

4.4. Late acceptance hill-climbing (LAHC)

LAHC is a single point meta-heuristic inspired room hill-climbing search proposed by Burke [50]. Unlike Hill-climbing, LAHC can escape local optimums by maintains a list of a given length L, which is a kind of memory unit. This list retains solutions of several iterations earlier for comparison with the current candidate solution. LAHC starts with a single feasible solution, and iteratively improves the solution in order to get a new improve one. Each time the candidate solution is compared with the last value of the list, L and if better, it is accepted. When the acceptance procedure is activated, the new cost is added at the beginning of the list, and the last element is deleted. The procedure is performed base on $v = I \ mod \ L$ formula, where L is the length of the frame, I is the ith iteration and v is the position. Figure 3 shows the LAHC procedure.

```
1.  Calculate initial cost function C(s)
2.  .for all k ∈ {0…L − 1} do Ĉ_k ← C(s)
3.  Assign the initial number of iteration I ← 0
4.  do until a stopping criterion is satisfied:
5.      Construct a candidate solution s*
6.      Calculate its cost function C(s*)
7.      v ← I mod L
8.      if C(s*) ≤ Ĉ_v or C(s*) ≤ C(s)
9.          then accept candidate (s ← s*)
10.     else
11.         accept candidate (s ← s)
12.     Insert cost value into the list Ĉ_v ← C(s)
13.     Increment the number of iteration I ← I+1
14. end do
```

Figure 3. Late acceptance hill-climbing search algorithm.

4.5. Great deluge algorithm (GDA)

Great deluge algorithm (GDA) is a local search algorithm developed by Dueck [51]. The inspiration of this algorithm originated from the behaviour that a hill climber seeks a higher place to avoid the rising water level during the deluge. Like SA, this algorithm devises a mechanism to avoid local optima by accepting the worst solution. However, SA uses a probabilistic function for accepting the worst solutions, whereas GDA uses a more deterministic approach for this purpose. It is also considered that GDA depends less on parameter tuning compared to SA. The only parameter in the GDA is decay rate, which is used for controlling the boundary or acceptance level. In the minimization problem, the initial boundary level (water level) usually starts with an initial solution. During the search, a new candidate solution is accepted if it is better than or equal to the current solution. However, the solution worse than the current one will be accepted if the quality of the candidate solution is less than or equal to a predefined boundary level B. The boundary level then is lowered by

subtracting a parameter called decay rate (ΔB). This parameter is vital because the speed of the search depends on the decay rate. Figure 4 describes the procedure of the GDA algorithm in a minimization context.

1. Set the initial solution S
2. Calculate initial cost function $f(S)$
3. Initial level $B_0 = f(S)$
4. $B = B_0$
5. Specify input parameter ΔB =?
6. **while** not some stopping condition do
7. Define neighbourhood $N(S)$
8. Randomly select the candidate solution $S^* \in N(S)$
9. **If** $f(S^*) \leq f(S)$ *or* $f(S^*) \leq B$ **then**
10. Accept S^*
11. Lower the level $B = B_0 - \Delta B$

Figure 4. Great deluge algorithm procedure.

4.6. Variable neighbourhood search (VNS)

Variable neighbourhood search (VNS), a local search descent method, was first introduced by Mladenović and Hansen [52]. VNS does not accept non-improving neighbours. The basic idea of VNS is based on changing the landscape of the problem using more than one neighbourhood searches. The algorithm has three basic steps: shaking, local search, and move. Initially, a set of neighbourhood structures and initial solutions is defined. In each iteration, an initial solution is shaken from the current neighbour N_k (x' is generated in current neighbour). A local search approach is then used to transform this x' solution to x'' solution. When $f(x'')$ is better than $f(x)$, current solution x will be replaced by the solution x'' , and the search starts over from the first neighbourhood $(k = 1)$. Otherwise, the algorithm uses the next neighbourhood $(k = k + 1)$. Several variations of VNS are found. Figure 5 presents a basic variable neighbourhood search approach.

1. **Input:** a set of neighbourhood structures N_k for $k = 1, \dots, k_{max}$ for shaking
2. $.x = x_0$; /∗ Generate the initial solution ∗/
3. **repeat**
4. $k = 1$;
5. **repeat**
6. Shaking: pick a random solution x' from the k^{th} neighbourhood $N_k(x)$ for x;
7. x'' =local search(x') ;
8. **if** $f(x'') < f(x)$ **then**
9. $x = x''$;
10. Continue to search with N_1 ; $k = 1$;
11. **otherwise** $k = k + 1$;
12. **until** $k = k_{max}$
13. **until** Stopping criteria
14. **output:** Best found solution

Figure 5. Variable neighborhood search algorithm.

5. MATERIALS AND METHODS

5.1 Hybridization of Graph heuristics

The first step is to construct initial feasible solutions for examination timetabling problem. We use LD, LWD, and LE, which are static ordering. Besides, we also hybridize each of LD, LWD, and LE with SD as dynamic ordering heuristics. It is observed that SD tends to perform better than the three heuristics on many occasions [18, 53]. However, at the very beginning of the solution construction, SD may not be efficient like LE, LWD, and LD due to most of the time slots being unoccupied, resulting in difficulties for SD in the appropriate ordering of examinations [17]. Here we describe three hybridized graph heuristics SD-LD, SD-LWD, and SD-LWD and provide an illustrative example for better understanding the procedure.

5.1.1 Definition

- SD-LD: It means ordering the examinations according to SD followed by LD and taking the most crucial examination from the top of the list for scheduling.
- SD-LWD: It indicates ordering the examinations according to SD followed by LWD and taking the most crucial examination from the top of the list for scheduling.
- SD-LE: It denotes ordering the examinations according to SD, followed by LE and taking the most crucial examination from the top of the list for scheduling.

5.1.2 An illustrative example

Conflicting of examinations and ordering procedure can be illustrated using the following examples. Consider a conflict matrix M consisting of 9 examinations $(e1, e2, e3, e4, e5, e6, e7, e8, e9)$ in Figure 6. Here an entry $M\ (e2, e7)$ has value 2. It means that there is a conflict between examination $e2$ and examination $e7$ and 2 students have taken these two courses. Similarly, the entry $M\ (e1, e4)$ has value 0, indicating no conflict between these two examinations (i.e., no common students have taken these two courses). In this way, other entities can be defined. Note that $M\ (e2, e7) = M(e7, e2)$, as the matrix is symmetrical and diagonal items in the matrix have zero values, meaning no conflict exists between two same examinations.

exams	e1	e2	e3	e4	e5	e6	e7	e8	e9
e1	0	1	2	0	2	2	3	0	1
e2	1	0	1	1	1	1	2	1	0
e3	2	1	0	0	1	1	2	0	1
e4	0	1	0	0	0	0	0	0	0
e5	2	1	1	0	0	1	3	1	0
e6	2	1	1	0	1	0	4	1	2
e7	3	2	2	0	3	4	0	2	2
e8	0	1	0	0	1	1	2	0	0
e9	1	0	1	0	0	2	2	0	0

Figure 6. Conflict matrix

Each row is considered a vector of the matrix is dedicated to a particular examination (i.e., $e1$), and its column values with non-zero are all conflicted examinations with that particular examination. For example, $e2, e3, e5, e6, e7, e9$ are conflicting with $e1$. For getting LD ordering, all the column examinations which conflict with each row of the matrix that is not zero are considered first, and then the number of conflicting exams is counted. This specifies LD value of the examination in this row. Now the LD ordering is obtained by sorting these LD values in decreasing order. Figure 7 (a) illustrates the LD ordering. Here $e2$ examination is top of the LD ordering as it has the maximum value 7, whereas $e4$ is at the bottom of the list because of its lower ordering value 1.

Exams	LD
e2	7
e6	7
e7	7
e1	6
e3	6
e5	6
e8	4
e9	4
e4	1

(a)

Exams	LWD
e7	18
e6	12
e1	11
e5	9
e3	8
e2	8
e9	6
e8	5
e4	1

(b)

Exams	LE
e7	6
e1	4
e3	4
e4	4
e6	4
e2	3
e5	3
e9	3
e8	2

(c)

Exams	SD	LD
e7	2	7
e5	3	6
e3	3	6
e9	3	4
e1	4	6
e8	4	4
e4	4	1
e6	5	7
e2	5	7

(d)

Exams	SD	LWD
e7	2	18
e5	3	9
e3	3	8
e9	3	6
e1	4	11
e8	4	5
e4	4	1
e6	5	12
e2	5	8

(e)

Exams	SD	LE
e7	2	6
e3	3	4
e5	3	3
e9	3	3
e1	4	4
e4	4	4
e8	4	2
e6	5	4
e2	5	3

(f)

Figure 7. Example of various graph heuristic orderings (a) Ordered by LD (b) Ordered by LWD (c) Ordered by LE (d) Ordered by SD-LD (e) Ordered by SD-LWD (f) Ordered by SD-LE.

In the case of LWD ordering, all column examinations that are conflicting with each row examination are collected. Next, the sum of all conflicting values of all column examinations in each row of the matrix produces LWD value of that row examination. Finally, when all the LWD values are arranged in decreasing order, LWD ordering is found. Figure 7 (b) describes LWD ordering where $e7$ is at the top of the list due to its largest value of 18 followed by $e6$ with the second largest value of 12.

LE ordering, however, considers student enrolment data and avoids conflict matrix for the ordering process. Examination with the largest enrolment of students is considered at the top of the list. For instance, the enrolment of students is like this: e1 has been taken by 4, e2 has been taken by 3, e3 has been taken by 4, e4 has been taken by 4, e5 has been taken by 3, e6 has been taken by 4, and e7 has been taken by 6 students. Arranging them in decreasing order based on enrolments, LE ordering of these examinations is obtained, which is shown in Figure 7 (c).

SD is a dynamic process that needs information about the current timetabling state. In a particular time, each unscheduled examination checks the number of available time slots for scheduling without violating hard constraints. This number indicates SD values of that examination. For example, if $e6$ has SD value 5, it means that $e6$ has 5 free time slots where it can be assigned. Unlike other orderings, SD ordering is obtained by sorting the unscheduled examinations in ascending order so that examination with the least number of available time slots gets the first priority for scheduling. From Figure 7 (d-f), it can be seen that $e7$ is at the top of the SD ordering list. This is because $e7$ has the least number of free time slots, only two-time slots available

for scheduling. Likewise, $e6$ is at the bottom of the list because of its five available time slots for the allocation of examination $e6$.

SD-LD, SD-LWD, SD-LE orderings are produced by hybridizing SD with other heuristic orderings in such a way that SD is employed for ordering examinations fast followed by employing other orderings. Figure 7(d), Figure 7(e), and Figure 7(f) indicate the ordering of SD-LD, SD-LWD, and SD-LE, respectively. In these cases, SD orders the examinations first, and then the adjacent heuristic is employed for ordering. For example, in SD-LWD ordering, it is observed that e7 is at the top of the list because it has the lowest SD value. It is observed that $e5, e3,$ and $e9$ have the same SD value, but they have different LWD values. If two or more examinations have the same SD value, then LWD is considered. Examination e5 comes first because its LWD value is higher than both $e5$ and $e3$. Therefore, the ordering of these three examinations will be $e5$ followed by $e3$ and then e9. Since SD is solely unable to order the examinations $e5, e3,$ and $e9$ properly, second time ordering (in this example LWD) assists in producing robust ordering.

5.2 Improvement with Trajectory search

In this step, the initial feasible solution is further improved by trajectory-based methods in order to produce a near-optimal solution(s). The initial solution for trajectory metaheuristic is calculated using a graph heuristic that produces the best solution during the construction phase. Five trajectory-based methods comprising of tabu search (TS), late acceptance hill-climbing search (LAHC), simulated annealing(SA), great deluge algorithm(GDA), and variable neighbourhood search (VNS) have been used during improvement phase.

5.3 Experimental setup

We have considered two commonly used benchmark datasets in examination timetabling research, which are Toronto and ITC2007 datasets, to assess the performance of our approach. We have used 12 instances of Toronto benchmark datasets and 8 instances of ITC2007 benchmark datasets.

Neighbourhood structure for Toronto datasets during the improvement phase is described as below:

- N_1: Move – an examination is selected randomly and moves it to a random time slot.
- N_2: Swap – Two examinations are selected randomly, and swapping is occurred between their time slots.
- N_3: Swap time slot – Two-time slots are selected randomly, and all examinations between the two-time slots are swapped.

The above three (3) neighbourhood structures are used during the improvement phase. However, a neighbourhood structure is only accepted that gives an improvement on the penalty value in each iteration.

The neighbourhood operations employed in the improvement phase for ITC2007 exam datasets are as follows:

- N_1: An examination is selected randomly and moves it to a random time slot and room
- N_2: Two examinations are selected randomly and swapping is occurred between their time slots and rooms
- N_3: An examination is selected and moves it to a different room within the same time slot
- N_4: Two random examinations are selected and move them to different time slots and rooms

During the improvement phase, a neighbourhood move from these neighbourhood structures is selected randomly and applied only if the solution is feasible; otherwise, a different neighbourhood move is selected. Besides, stopping criteria for Toronto and ITC2007 are set to 30 min and one hour, respectively. Finally, each experiment is run 30 individuals using different random seeds to obtain computational results.

The programs were implemented in Java (Java SE 7) and performed on Intel Core-i7 PCs with 8 GB RAM running Windows 7 Professional SP3. For getting the appropriate values of the parameters in meta-heuristic algorithms, some preliminary experiment has been conducted. Table 4 shows the details of the parameters used for the study.

Table 4. Parameter settings

Name	Parameter	Value
SA	Cooling rate	0.1
	Temperature	5000
LAHC	List size	500
GDA	Decay rate	0.1
VNS	Neighbourhood, K	3 (for Toronto) 4 (for itc2007)
	Local search	Hill-climbing
TS	Tabu list size	100

6. RESULTS AND DISCUSSION

A comparative study of different graph colouring algorithms on the Toronto datasets for constructing initial solutions is highlighted in Table 5. In this table, the best and the corresponding average value produced by graph colouring algorithms for each instance is highlighted. Note that, from now to subsequent tables, the best results obtained from all the approaches for each problem instance are highlighted in the table with bold font, while '–' indicates no solution obtained. As it is observed from the table, SD-LD achieved the best results on 5 instances (car-f-92, kfu-s-93, rye-s-93, ute-s-92, yor-f-83), whereas SD-LWD outperformed others on 4 instances (ear-f-83, lse-f-91, sta-f-83, tre-s-92). The rest of the 3 instances SD-LE produced the best results. It is also noticed that without the hybridization of SD, individually 3 heuristics LD, LE, and LWD could not produce the best solutions for any of the instances.

Table 5. Different graph coloring algorithms on the Toronto datasets for constructing solutions

Instances	LD		LE		LWD		SD-LD		SD-LE		SD-LWD	
	Best	Avg	Best	Avg	Best	Avg	Best	Avg	Best	Avg	Best	Avg
car-s-91	8.70	8.97	8.59	9.14	8.56	9.07	8.48	8.90	**8.33**	8.80	8.52	8.90
car-f-92	7.88	7.98	7.45	7.67	7.44	7.77	**7.00**	7.44	7.25	7.56	7.35	7.50
ear-f-83	54.23	57.99	54.40	58.11	54.40	58.11	54.14	57.76	53.05	57.73	**52.35**	57.17
hec-s-92	17.51	20.40	17.24	20.24	16.94	20.92	16.95	20.18	**16.21**	20.37	16.37	20.51
kfu-s-93	24.99	28.3	25.01	28.47	24.98	28.53	**23.68**	27.99	23.85	28.43	24.19	28.21
lse-f-91	19.78	24.14	20.01	24.47	19.42	25.17	19.36	23.89	19.71	24.04	**18.83**	24.09
rye-s-93	20.17	21.92	19.33	20.98	18.75	20.79	**18.28**	20.61	19.15	20.72	18.62	20.40
sta-f-83	166.77	180.36	169.13	178.31	171.81	179.97	168.35	178.36	167.95	178.50	**166.43**	177.08
tre-s-92	12.53	13.34	12.40	13.32	12.46	13.30	12.39	13.23	12.35	13.20	**12.07**	13.17
uta-s-92	5.78	6.18	5.71	6.20	5.54	6.19	5.68	6.17	**5.53**	6.17	5.70	6.02
ute-s-92	-	-	-	-	39.9	43.1	**38.03**	42.14	38.77	42.83	38.15	42.98
yor-f-83	51.12	53.77	51.11	54.16	51.58	53.60	**49.80**	53.67	9.82	53.63	51.02	54.02

Table 6 shows the comparison of the best and average results of six graph colouring approaches on ITC2007 datasets. It is observed that 4 out of 8 instances (Exam_2, Exam_4, Exam_5, Exam_8), SD-LD reported the best results, while SD-LWD reported the best results for the other four instances. LD, LE, and LWD, however, could not produce any solution for Exam_4. Besides, for the rest of the instances, they were not able to perform well than their hybridization with SD counterparts, which indicates the strength of the hybridization approaches for solution construction.

Table 6. Different graph coloring algorithms on the ITC2007 datasets for constructing solutions

Instances	LD		LE		LWD		SD-LD		SD-LE		SD-LWD	
	Best	Avg	Best	Avg	Best	Avg	Best	Avg	Best	Avg	Best	Avg
Exam_1	26540	30796.83	27967	30241.30	28135	30555.47	26245	29742.13	26578	29668.00	**25989**	29963.90
Exam_2	15567	21849.59	14887	19859.56	14375	1876.58	**11684**	17234.34	12345	19345.34	12257	18856.32
Exam_3	45956	54869.95	45898	53639.47	45778	5534.69	44678	52568.89	43968	51849.53	**43588**	52879.78
Exam_4	-	-	-	-	-	-	**41702**	48616.27	43853	50240.50	43042	514321.67
Exam_5	64856	73945.48	65438	73172.29	65485	72394.34	**63895**	70367.53	64265	71567.56	63885	71767.78
Exam_6	45588	54737.67	46390	54565.17	45390	54165.17	45300	54375.17	45195	54263.67	**44160**	52832.50
Exam _7	35789	43667.36	36158	4266.84	35338	43854.47	34567	42385.42	34857	41945.88	**33557**	42475.63
Exam _8	44576	54837.85	44978	53685.64	45069	53857.46	**43866**	52475.26	44012	51845.79	44234	52476.46

Table 7 shows the performance of different trajectory algorithms employed on Toronto datasets in obtaining quality solutions. The best and average values are shown for each instance. It is apparent from the table that GDA outperformed other algorithms because, in 7 out of 12 cases, it resulted in the best results. SA is the second best algorithm that reported the best values for four instances. There are two instances (kfu-s-93 and ute-s-92) in which LAHC produced the best solutions. However, VNS and TS could not produce better results in comparison with GDA, LAHC, and SA.

Table 7. Different Trajectory meta-heuristics on the Toronto datasets for improving solutions

Instances	TS		SA		LAHC		GDA		VNS	
	Best	Avg	Best	Avg	Best	Avg	Best	Avg	Best	Avg
car-s-91	5.58	5.83	5.57	5.78	5.55	5.77	**5.54**	5.76	5.61	5.85
car-f-92	4.58	4.93	4.61	4.82	4.57	4.93	**4.57**	4.83	4.72	4.92
ear-f-83	35.57	37.47	**33.68**	36.16	33.72	36.56	34.01	35.96	37.65	38.62
hec-s-92	10.73	11.76	10.55	11.53	10.60	11.25	**10.53**	11.32	10.61	11.78
kfu-s-93	14.89	17.01	14.95	16.26	**14.79**	15.57	14.83	15.69	15.21	16.12
lse-f-91	11.23	12.12	11.15	12.14	10.94	11.02	**10.85**	11.12	10.95	11.23
rye-s-93	11.31	11.51	11.41	11.48	11.27	11.56	**11.24**	11.54	11.45	11.55
sta-f-83	157.47	157.78	**157.39**	157.72	157.41	157.76	157.43	157.68	157.56	157.69
tre-s-92	8.64	8.74	**8.24**	8.50	8.38	8.65	8.25	8.57	8.27	8.51
uta-s-92	3.28	3.49	**3.22**	3.29	3.28	3.39	3.27	3.41	3.25	3.52
ute-s-92	27.01	27.98	27.14	27.97	**26.98**	28.07	**26.98**	28.10	27.32	28.23
yor-f-83	35.67	36.78	35.53	26.78	35.66	36.91	**35.51**	36.71	35.89	38.01

Table 8 highlights the best and average results of the instances of ITC2007 datasets when the trajectory metaheuristic approaches are employed for improving solution quality. It is observed that the performance of the algorithms under investigation here produces comparable results. However, a closer look reveals that GDA is the most successful in producing quality solutions. It produced the best results for Exam_1, Exam_3, Exam_5, and Exam_6. The next best metaheuristic is LAHC, which produced the best results for three datasets (Exam_6, Exam_7, Exam_8). The rest of the instances (Exam_2 and Exam_4) had the best solutions with SA approach. Results also reveal that, during the improvement process, TS and VNS are not as robust as the rest of the approaches in terms of producing the best solutions.

Table 8. Different trajectory meta-heuristics on the ITC2007 datasets for improving solutions

Instances	TS		SA		LAHC		GDA		VNS	
	Best	Avg	Best	Avg	Best	Avg	Best	Avg	Best	Avg
Exam_1	10475	11576.49	10875	11674.78	9791	10465.34	**8571**	10710.93	10123	11282.64
Exam_2	996	1170.77	**895**	1172.56	984	1165.33	923	1134.56	989	1234.74
Exam_3	23052	27456.67	24635	27375.48	24012	27846.95	**22345**	26955.67	22175	26487.38
Exam_4	26194	28880.72	**25658**	27947.58	26547	28012.47	25975	27859.48	26012	28210.45
Exam_5	7684	7984.94	7584	7845.74	7475	7785.75	**7415**	7877.17	7748	8012.34
Exam_6	30018	31237.56	3028	30457.34	**29960**	30985.74	**29960**	30855.74	30123	30894.05
Exam _7	9465	9849.56	9101	9584.89	**8899**	9457.41	9399	9757.41	9123	9763.47
Exam _8	11893	12547.07	11354	12565.97	**11345**	12847.34	11845	12461.24	11924	13127.56

Tables 9 and 10 show the best results obtained in our experiment and a selection of the best results available in the literature on Toronto and ITC2007 datasets, respectively. As shown in Table 9, our results are better than both Carter et al. [14] and Pillay and Banzhaf [54] for 8 problem instances, Sabar et al. [17] for 10 problem instances, Abdul Rahman et al. [18] for 7 problem instances, Caramia et al. [55] for 6 problem instances, and Turabieh and Abdullah [56] for 4 problem instances. Finally, from Table 10, it is observed that our method obtains the best results on five out of 13 instances compared to Pillay [57]. Besides, our results are better than Atsuta et al. [58] and Abdul Rahman et al. [18] for 3 and 2 problem instances, respectively. De Smet [59] produced better results than our method, but they could not produce feasible solutions for three instances. Overall, our results are competitive with other approaches in the literature.

Table 9. Comparison of our best results with results of state-of-the-art approaches on Toronto datasets

Instances	Our results	Carter et al. [14]	Abdul Rahman et al. [18]	Turabieh and Abdullah [56]	Caramia et al. [55]	Pillay and Banzhaf [54]	Sabar et al. [17]
car-s-91	5.54	7.10	5.12	4.8	6.6	4.97	5.14
car-f-92	4.57	6.20	4.41	4.1	6.0	4.28	4.70
ear-f-83	33.68	36.40	36.91	34.92	29.3	35.86	37.86
hec-s-92	10.53	10.80	11.31	10.73	9.2	11.85	11.90
kfu-s-93	14.79	14.00	14.75	13.0	13.8	14.62	15.30
lse-f-91	10.85	10.5	11.41	10.01	9.6	11.14	12.33
rye-s-93	11.24	7.3	9.61	9.65	6.8	9.65	10.71
sta-f-83	157.39	161.5	157.52	158.26	158.2	158.33	160.12
tre-s-92	8.24	9.6	8.76	7.88	9.4	8.48	8.32
uta-s-92	3.22	3.5	3.54	3.2	3.5	3.40	3.88
ute-s-92	26.98	25.8	26.25	26.11	24.4	28.88	32.67
yor-f-83	35.51	41.7	39.67	36.22	36.2	40.74	40.53

Table 10. Comparison of our best results with results of state-of-the-art approaches on ITC2007 datasets

Instances	Our results	Pillay [57]	Atsuta et al. [58]	De Smet [59]	Abdul Rahman et al. [18]
Exam_1	8571	12035	8006	6670	5231
Exam_2	895	3074	3470	623	433
Exam_3	22345	15917	18622	-	9265
Exam_4	25658	23582	22559	-	17787
Exam_5	7415	6860	4714	3847	3083
Exam_6	29960	32250	29155	27815	26060
Exam _7	8899	17666	10473	5420	10712
Exam _8	11345	16184	14317	-	12713

7. CONCLUSIONS

Academic institutions face challenges when scheduling exams into a limited number of timeslots and rooms. This is because constructing good quality timetables is a computationally expensive task. In this paper, we have examined different graph colouring heuristics and trajectory metaheuristics to solve the examination timetabling problem. Feasible solutions are constructed using six different graph colouring algorithms, which are LD, LWD, LE, SD-LE, SD-LWD, SD-LE. Among the graph heuristics that produce the best solution is considered to produce the initial solution of trajectory metaheuristics. Next, this solution is improved separately using five trajectory meta-heuristics comprising TS, SA, LAHC, GDA, and VNS. We have tested the approach on Toronto and ITC2007 examination timetable datasets. Based on the experimental results, we have drawn several conclusions.

During feature construction, hybridization of graph colorant algorithms (i.e., SD-LE, SD-LWD, and SD-LE) perform better than their non-hybridization counterparts for both of the datasets. The reason is that the dynamic nature of SD can effectively select more sophisticated and suitable exams early in the scheduling list during construction. On the other hand, in the optimization step, in general, GDA has proved to be more effective in finding quality solutions for ITC2007 datasets then the SA, LAHC, TS, and VNS in that order. In the case of Toronto datasets, GDA also performs better than others, followed by LAHC, SA, VNS, and TS. The main reason for obtaining better performance with GDA, SA, and LAHC is that they can properly explore and exploit the search space, which results in fewer occasions of trapping into local optima. Finally, we have been able to produce competitive results compared to other state-of-the-art methods.

Future studies should include applying graph heuristics with population metaheuristics on the examination timetabling problem. Another possible extension of this research would be to solve other scheduling problems such as course timetabling problem.

ACKNOWLEDGMENTS

This work was supported in part by grants from University Malaysia Pahang (Grant Id: RDU1703212).

REFERENCES

[1] N. Pillay and R. Qu, "Examination Timetabling Problems," in Hyper-Heuristics: Theory and Applications, ed: Springer, 2018, pp. 75-82.
[2] R. Lewis, "A survey of metaheuristic-based techniques for University Timetabling problems," OR Spectrum, vol. 30, pp. 167-190, 2008.
[3] M. N. M. Kahar and G. Kendall, "The examination timetabling problem at Universiti Malaysia Pahang: Comparison of a constructive heuristic with an existing software solution," European Journal of Operational Research, vol. 207, pp. 557-565, Dec 1 2010.
[4] N. Leite, R. Neves, N. Horta, F. Melício, and A. C. Rosa, "Solving a Capacitated Exam Timetabling Problem Instance Using a Bi-objective NSGA-II," in Computational Intelligence: International Joint Conference, IJCCI 2012 Barcelona, Spain, October 5-7, 2012 Revised Selected Papers, K. Madani, D. A. Correia, A. Rosa, and J. Filipe, Eds., ed Cham: Springer International Publishing, 2015, pp. 115-129.
[5] N. R. Sabar, M. Ayob, G. Kendall, and Q. Rong, "Grammatical Evolution Hyper-Heuristic for Combinatorial Optimization Problems," Evolutionary Computation, IEEE Transactions on, vol. 17, pp. 840-861, 2013.
[6] N. Leite, F. Melício, and A. C. Rosa, "A Hybrid Shuffled Frog-Leaping Algorithm for the University Examination Timetabling Problem," in Computational Intelligence: Revised and Selected Papers of the International Joint Conference, IJCCI 2013, Vilamoura, Portugal, September 20-22, 2013, K. Madani, A. Dourado, A. Rosa, J. Filipe, and J. Kacprzyk, Eds., ed Cham: Springer International Publishing, 2016, pp. 173-188.
[7] N. S. Othman and F. Mashhod, "Graph Colouring and Clustering Heuristic Approach for Minimizing Examination Duration: A Case Study," IBIMA Business Review, vol. 2012, pp. 1-9, 2012.

[8] A. K. Mandal and M. N. M. Kahar, "Solving examination timetabling problem using partial exam assignment with great deluge algorithm," in 2015 International Conference on Computer, Communications, and Control Technology (I4CT), 2015, pp. 530-534.
[9] M. Alzaqebah and S. Abdullah, "Hybrid bee colony optimization for examination timetabling problems," Computers & Operations Research, vol. 54, pp. 142-154, Feb 2015.
[10] A. Muklason, A. J. Parkes, E. Özcan, S. N. Kingston, B. McCollum, and P. McMullan, "Hyper-heuristics for Solving a Multi-objective Examination Timetabling Problem," in Proceedingsofthe12thInternationalConferenceonthePracticeandTheoryofAuto-matedTimetabling, Austria, 2018.
[11] S. Larabi Marie-Sainte, "A survey of Particle Swarm Optimization techniques for solving university Examination Timetabling Problem," Artificial Intelligence Review, vol. 44, pp. 537-546, 2015/07/12 2015.
[12] S. Abdullah, H. Turabieh, B. McCollum, and P. McMullan, "A hybrid metaheuristic approach to the university course timetabling problem," Journal of Heuristics, vol. 18, pp. 1-23, Feb 2012.
[13] R. Qu, E. Burke, B. McCollum, L. Merlot, and S. Lee, "A survey of search methodologies and automated system development for examination timetabling," Journal of Scheduling, vol. 12, pp. 55-89, Feb 2009.
[14] M. W. Carter, G. Laporte, and S. Y. Lee, "Examination timetabling: Algorithmic strategies and applications," Journal of the Operational Research Society, vol. 47, pp. 373-383, 1996.
[15] B. McCollum, P. McMullan, E. K. Burke, A. J. Parkes, and R. Qu, "The second international timetabling competition: Examination timetabling track," University of Nottingham, Technical Report QUB/IEEE/Tech/ITC2007/Exam/v4. 0/17, 2007.
[16] B. McCollum, P. McMullan, A. J. Parkes, E. K. Burke, and R. Qu, "A new model for automated examination timetabling," Annals of Operations Research, vol. 194, pp. 291-315, Apr 2012.
[17] N. R. Sabar, M. Ayob, R. Qu, and G. Kendall, "A graph coloring constructive hyper-heuristic for examination timetabling problems," Applied Intelligence, vol. 37, pp. 1-11, Jul 2012.
[18] S. Abdul Rahman, A. Bargiela, E. K. Burke, E. Özcan, B. McCollum, and P. McMullan, "Adaptive linear combination of heuristic orderings in constructing examination timetables," European Journal of Operational Research, vol. 232, pp. 287-297, 1/16/ 2014.
[19] S. Abdullah, E. K. Burke, and B. Mccollum, "An investigation of variable neighbourhood search for university course timetabling," in The 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2005), 2005, pp. 413-427.
[20] A. K. Mandal and M. N. M. Kahar, "Combination of graph heuristic with hill climbing search for solving capacitated examination timetabling problem," in 2015 4th International Conference on Software Engineering and Computer Systems (ICSECS), 2015, pp. 118-123.
[21] A. K. Mandal and M. N. M. Kahar, "Solving examination timetabling problem using partial exam assignment with hill climbing search," in 2015 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), 2015, pp. 84-89.
[22] T. C. Pais and P. Amaral, "Managing the tabu list length using a fuzzy inference system: an application to examination timetabling," Annals of Operations Research, vol. 194, pp. 341-363, Apr 2012.
[23] S. Abdullah, H. Turabieh, B. McCollum, and P. McMullan, "A Tabu-Based Memetic Approach for Examination Timetabling Problems," in Rough Set and Knowledge Technology. vol. 6401, J. Yu, S. Greco, P. Lingras, G. Wang, and A. Skowron, Eds., ed: Springer Berlin Heidelberg, 2010, pp. 574-581.
[24] M. Battistutta, A. Schaerf, and T. Urli, "Feature-based tuning of single-stage simulated annealing for examination timetabling," Annals of Operations Research, pp. 1-16, 2015.
[25] T. L. June, J. H. Obit, Y.-B. Leau, and J. Bolongkikit, "Implementation of Constraint Programming and Simulated Annealing for Examination Timetabling Problem," in Computational Science and Technology, ed: Springer, 2019, pp. 175-184.
[26] C. Gogos, P. Alefragis, and E. Housos, "An improved multi-staged algorithmic process for the solution of the examination timetabling problem," Annals of Operations Research, vol. 194, pp. 203-221, Apr 2012.
[27] E. K. Burke and Y. Bykov, "The late acceptance hill-climbing heuristic," Technical report CSM-192 Computing Science and Mathematics, University of Stirling, UK, University of Stirling, UK, Technical report CSM-192, 2012.
[28] M. Alzaqebah and S. Abdullah, "An adaptive artificial bee colony and late-acceptance hill-climbing algorithm for examination timetabling," Journal of Scheduling, vol. 17, pp. 249-262, 2014/06/01 2014.
[29] Y. Bykov and S. Petrovic, "A Step Counting Hill Climbing Algorithm applied to University Examination Timetabling," Journal of Scheduling, pp. 1-14, 2016.
[30] M. N. M. Kahar and G. Kendall, "A great deluge algorithm for a real-world examination timetabling problem," Journal of the Operational Research Society, vol. 66, pp. 16–133, 2013.
[31] H. Turabieh and S. Abdullah, "An integrated hybrid approach to the examination timetabling problem," Omega-International Journal of Management Science, vol. 39, pp. 598-607, Dec 2011.
[32] N. Pillay and W. Banzhaf, "An informed genetic algorithm for the examination timetabling problem," Applied Soft Computing, vol. 10, pp. 457-467, Mar 2010.
[33] M. Hosny and M. Al-Olayan, "A mutation-based genetic algorithm for room and proctor assignment in examination scheduling," in Science and Information Conference (SAI), 2014, 2014, pp. 260-268.
[34] M. Alinia Ahandani, M. T. Vakil Baghmisheh, M. A. Badamchi Zadeh, and S. Ghaemi, "Hybrid particle swarm optimization transplanted into a hyper-heuristic structure for solving examination timetabling problem," Swarm and Evolutionary Computation, vol. 7, pp. 21-34, 12// 2012.

[35] S. L. Marie-Sainte, "A new hybrid particle swarm optimization algorithm for real-world university examination timetabling problem," in 2017 Computing Conference, 2017, pp. 157-163.
[36] A. L. Bolaji, A. T. Khader, M. A. Al-Betar, and M. A. Awadallah, "A hybrid nature-inspired artificial bee colony algorithm for uncapacitated examination timetabling problems," Journal of Intelligent Systems, vol. 24, pp. 37-54, 2015.
[37] E. Ozcan and E. Ersoy, "Final exam scheduler - FES," in IEEE Congress on Evolutionary Computation, 2005, pp. 1356-1363
[38] S. Abdullah and H. Turabieh, "On the use of multi neighbourhood structures within a Tabu-based memetic approach to university timetabling problems," Information Sciences, vol. 191, pp. 146-168, May 15 2012.
[39] Y. Lei, J. Shi, and Z. Yan, "A memetic algorithm based on MOEA/D for the examination timetabling problem," Soft Computing, vol. 22, pp. 1511-1523, 2018.
[40] N. Leite, C. M. Fernandes, F. Melicio, and A. C. Rosa, "A cellular memetic algorithm for the examination timetabling problem," Computers & Operations Research, vol. 94, pp. 118-138, 2018.
[41] N. Pillay, "A review of hyper-heuristics for educational timetabling," Annals of Operations Research, vol. 239, pp. 3-38, Apr 2016.
[42] K. Anwar, A. T. Khader, M. A. Al-Betar, and M. A. Awadallah, "Harmony Search-based Hyper-heuristic for examination timetabling," in 9th IEEE International Colloquium on Signal Processing and its Applications (CSPA), 2013, pp. 176-181.
[43] P. Demeester, B. Bilgin, P. De Causmaecker, and G. Vanden Berghe, "A hyperheuristic approach to examination timetabling problems: benchmarks and a new problem from practice," Journal of Scheduling, vol. 15, pp. 83-103, Feb 2012.
[44] I. H. Osman and G. Laporte, "Metaheuristics: A bibliography," Annals of Operations Research, vol. 63, pp. 511-623, 1996/10/01 1996.
[45] B. Hussin, A. S. H. Basari, A. S. Shibghatullah, S. A. Asmai, and N. S. Othman, "Exam timetabling using graph colouring approach," in IEEE Conference on Open Systems (ICOS), Lankawi, Malaysia, 2011, pp. 133-138.
[46] D. Marx, "Graph colouring problems and their applications in scheduling," Electrical Engineering, vol. 48, pp. 11-16, 2004.
[47] P. Galinier, J.-P. Hamiez, J.-K. Hao, and D. Porumbel, "Recent Advances in Graph Vertex Coloring," in Handbook of Optimization: From Classical to Modern Approach, I. Zelinka, V. Snášel, and A. Abraham, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 505-528.
[48] F. Glover, "Future paths for integer programming and links to artificial intelligence," Computers & operations research, vol. 13, pp. 533-549, 1986.
[49] S. Kirkpatrick and M. Vecchi, "Optimization by simmulated annealing," science, vol. 220, pp. 671-680, 1983.
[50] E. K. Burke and Y. Bykov, "A late acceptance strategy in hill-climbing for exam timetabling problems," in The 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2008) Montreal, Canada, 2008.
[51] G. Dueck, "New optimization heuristics: the great deluge algorithm and the record-to-record travel," Journal of Computational physics, vol. 104, pp. 86-92, 1993.
[52] N. Mladenović and P. Hansen, "Variable neighborhood search," Computers & Operations Research, vol. 24, pp. 1097-1100, 11// 1997.
[53] A. Soghier and R. Qu, "Adaptive selection of heuristics for assigning time slots and rooms in exam timetables," Applied Intelligence, vol. 39, pp. 438-450, Sep 2013.
[54] N. Pillay and W. Banzhaf, "A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem," European Journal of Operational Research, vol. 197, pp. 482-491, Sep 1 2009.
[55] M. Caramia, P. Dell'Olmo, and G. F. Italiano, "Novel local-search-based approaches to university examination timetabling," Informs Journal on Computing, vol. 20, pp. 86-99, Win 2008.
[56] H. Turabieh and S. Abdullah, "A Hybrid Fish Swarm Optimisation Algorithm for Solving Examination Timetabling Problems," in Learning and Intelligent Optimization. vol. 6683, C. C. Coello, Ed., ed: Springer Berlin Heidelberg, 2011, pp. 539-551.
[57] N. Pillay, "A developmental approach to the examination timetabling problem," presented at the The 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2008), 2008.
[58] M. Atsuta, K. Nonobe, and T. Ibaraki, "ITC-2007 Track2: an approach using general CSP solver," 2008.
[59] G. De Smet, "ITC2007—examination track," in The 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, 2008.