Development of an Interactive Tool based on Combining Graph Heuristic with Local Search for Examination Timetable Problem

Ashis Kumar Mandal
Faculty of Software and Information Science
Iwate Prefectural University
Iwate, Japan

***Abstract*—Every university faces a lot of challenges to solve the examination timetabling problem because the problem is NP-hard and contains numerous institutional constraints. Although several attempts have been taken to address the issue, there are scarcities of interactive and automated tools in this domain that can schedule exams effectively by considering institutional resources, different constraints, and student enrolment in courses. This paper presents the development of a system as a graphical and interactive tool for examination timetabling problem. To develop the system, combining graph coloring heuristic and local search meta-heuristic algorithms are employed. The graph heuristic ordering is incorporated for constructing initial solution(s), whereas the local search meta-heuristic algorithms are used to produce quality exam timetables. Different constraints and objective functions from ITC2007 exam competition rules are adopted, as it is a complex real word exam timetabling problem. Finally, the system is tested on the ITC2007 exam benchmark dataset, and test results are presented. The main aspect of the system is to deliver an easy-to-handle tool that can generate quality timetables based on institutional demands and smoothly manage several key components. These components are collecting data associated with the enrolment of students in exams, defining hard and soft constraints, and allocating times and resources. Overall, this software can be used as a commercial scheduler in order to provide institutions with automated, accurate, and quick exam timetable.**

Keywords*—Examination timetable; graph heuristic; local search meta-heuristic; ITC2007 exam dataset; interactive tool; NP-hard problem

I. Introduction

Solving the examination timetable of an academic institution induces lots of complicacies due to the intractable nature of the problem. That is, managing different constraints and producing a quality exam timetable, which meets the demand of the institution, often computationally expensive and laborious task. The procedure is so complicated that human schedulers also struggle to produce even a simple feasible solution. Some viable approaches are operations research (OR) and artificial intelligence (AI) techniques that handle the problem often by mathematical programming as well as different heuristics, meta-heuristic, and hyper-heuristic algorithms [1]. Some of these procedures are constraint programming [2], integer programming [3], graph heuristics [4], great deluge algorithm [5], hill-climbing [6], tabu search [7], simulated annealing [8], genetic algorithm [9], particle swarm optimization [10], artificial bee colony algorithm [11] and memetic algorithm [12].

It is frequently observed that academic institutions rely on traditional manual approaches, which take considerable time on managing the vast amount of student registration data, conflicting exams, as well as the violation of constraints for producing a feasible timetable. Although numerous approaches have been proposed in the literature for creating quality timetables, the work on interactive software for timetabling problems is limited. In the examination timetabling survey, Qu et al. [13] emphasize reducing the gap between research and practice and highlight the importance of automatic interactive examination timetable tools for reducing the significant workload of timetabling staff. There are some software solutions proposed for university timetabling problems so that the user can easily interact with the timetable generations. Piechowiak and Kolski [14] developed interactive tools for supporting University of Valenciennes and Hainaut-cambresis (UVHC) university timetabling. The aim was to develop an open, generic tool that employs distributed architecture for cooperative scheduling and supports users to monitor modeling of time, resource, university activities, and constraints for producing a quick feasible solution. Thomas et al. [15] proposed a visual interface tool that aids users to quickly find a bottleneck situation and guide the scheduling system towards a feasible solution. Ayob et al. [16] proposed an intelligent examination timetabling software. The aim was to develop an intelligent commercial scheduler that can replace human decision-makers as well as produce high-quality solutions for University Kebangsaan Malaysia (UKM) examination timetabling problem. Chunbao and Nu [17] developed an efficient exam scheduling system (IIEESS v1.0) that avoids the traditional direct-clash-checking approach and schedules a large number of exam papers within a few minutes. Another recent software solution is solving the University of Toulouse examination timetabling problem using integer linear programming [18]. The authors claimed that the tool can produce quality solutions automatically and give some flexibility to choose input data and constraints.

Although the above tools are viable in producing quality exam timetabling, most of them emphasize solely on automation rather than interaction with users. Besides, interactive tools for generating student conflicts performed by open registration, simultaneous execution of different optimization processes, and handling more complex real-world problems like ITC2007

exam dataset are some issues that have been less highlighted.

This paper has proposed an interactive tool for addressing the examination timetabling issue for universities. The system initially produces conflicting exams automatically from course enrollment data. Then it facilitates in managing various hard and soft constraints and allocating times and resources. Based on that configuration, the initial solution module, which uses a saturation degree graph heuristic (SD), produces a feasible solution. Users can monitor the solution quality with different configurations and even further improve the solution vector with employing an improvement module. This module contains three local search algorithms: Great Deluge Algorithm (GDA), Simulated Annealing (SA), and Late Acceptance Hill Climbing (LAHC). A user can select different algorithms, tune parameters, and inspect the progression of the solution. In addition, another facility is the execution of concurrent run with selected local search algorithms. The improvement phase finally produces a solution vector along with a quality metric within specific stopping criteria. The proposed system has been tested successfully using ITC2007 exam benchmark dataset, which covers the majority of the hard and soft constraints of many real exam timetables. The goal in this paper is to provide an effective interactive examination timetable software such that it can generate computationally inexpensive quality timetable solutions and reduce both context-dependencies and involvement of human expertise as much as possible.

The rest of this paper is organized as follows: Section II describes the examination timetabling problem formulation. Section III highlights the algorithms employed for building the system. The proposed system architecture and software component for addressing examination timetabling problems have been presented in Section IV. Section V presents simulation results and discussion. Finally, some conclusions are drawn in Section VI.

II. Problem Description and Formulations

An examination timetabling is a scheduling problem where a set of examinations is allocated into a limited number of time slots and rooms subject to a set of constraints. Generally, two different types of constraints encompassing hard constraints, and soft constraints must be addressed. Satisfying all hard constraints leads to a feasible solution, whereas the minimization of soft constants results in a quality solution. Frequently these soft constraints are associated with objective functions. All hard and soft constraints vary from one institution to another based on institutional requirements and resources.

Examination timetable problems can be categorized as capacitated and un-capacitated problems. In an un-capacitated branch, room capacity is not considered. In a capacitated variant, however, room capacity is considered as a hard constraint. For instance, Toronto dataset is a un-capacitated problem, whereas the Second International Timetabling Competition (ITC2007) exam dataset is a capacitated problem [13].

In this section, ITC2007 exam dataset is described here because it is the most recent real-world examination timetabling problem, which has lots of hard and soft constraints. Besides, the proposed system has been developed with ITC2007 exam timetabling problem in mind. ITC2007 exam dataset consists of eight problem instances. The comprehensive characteristics such as number of students, number of exams, number of slots, number of rooms, period hard constraints, room hard constraints, and conflict density are presented in Table I.

All hard and soft constraints involved with the examination timetabling problem reported in ITC2007 exam dataset are given below:

Hard Constraints

- H1. Any student cannot sit more than one exam at the same time.
- H2. The exam capacity should not exceed room capacity.
- H3. The exam length should not violate the period length.
- H4. Three ordering of exams 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 at the same period.
- H5. Room exclusiveness must be maintained. For example, an exam i must take place only in room number 206.

Soft Constraints

- S1. Two exams in a row(C_s^{2R}): Avoid the number of occasions where a student sits consecutive exams on the same day.
- S2. Two exams in a day (C_s^{2D}): Avoid the number of occasions where a student sits two exams in a day. Note that when exams are one after another, this is counted as Two Exams in a Row for avoiding duplication.
- S3. Spreading of exams (C_s^{PS}): Exams should be spread as evenly as possible over time periods.
- S4. Mixed duration (C^{NMD}): Avoid the number of occasions where exams with different durations are scheduled into the same room.
- S5. Scheduling of larger exams (C^{FL}): Avoid the number of occasions where the largest exams are assigned later in the timetable.
- S6. Room penalty:(C^R) Avoid the number of occasions where certain rooms with an associated penalty are used for scheduling.
- S7. Period penalty (C^P): Avoid the number of occasions where certain periods with an associated penalty are used for scheduling.

The objective function is formularized as in Eq. 1. It attempts to minimize the violation of soft constraints (penalty) as much as possible for producing good quality solutions without violating the hard constraints.

TABLE I: Features of ICT2007 exam dataset

Instances	No. of students	No. of exams	No. of slots	No. of rooms	Period hard constraints	Room hard constraints	conflict density
Exam_1	7,891	607	54	7	12	0	5.05%
Exam_2	12,743	870	40	49	12	2	1.17%
Exam_3	16,439	934	36	48	170	15	2.62%
Exam_4	5,045	273	21	1	40	0	15.00%
Exam_5	9,253	1018	42	3	27	0	0.87%
Exam_6	7,909	242	16	8	23	0	6.16%
Exam_7	14,676	1096	80	15	28	0	1.93%
Exam_8	7,718	598	80	8	20	1	4.55%

$$min \sum_{s \in S} (W^{2R}C_s^{2R} + W^{2D}C_s^{2D} + W^{PS}C_s^{PS}) + W^{NMD}C^{NMD} + W^{FL}C^{FL} + C^R + C^P \quad (1)$$

In this equation, W (with different subscriptions) stands for the related weight for each of the soft constraints, and S indicates a set of students. Table II shows weights of ITC2007 exam dataset. Note that associated weights are not included in C^P and C^R in the equation as these associated weights are already added in the definition. Explaining all constraints, instances, mathematical models of the ITC2007 exam tracks are so wordy that details explanation will be found in [19] and the website at http://www.cs.qub.ac.uk/itc2007.

TABLE II: Weights of ITC2007 exam dataset

Instances	W^{2D}	W^{2R}	W^{PS}	W^{NMD}	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

III. Overview of Algorithms

A. Graph Heuristics

A simple Examination timetabling can be represented as a graph coloring problem. It is an undirected graph comprising a set of n vertices and a set of edges E, with vertices indicating exams while exams with common students indicate an edge. For example, If Exam_1 and Exam_2 have a common student, there will be an edge E. There are a predefined limited number of colors that signify time slots. The exams have to be assigned into time slots (i.e., coloring the graph) in such a way that no exams with a common student have the same timeslots (i.e., color). 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. Various graph heuristic techniques measure examination difficulty. These are the largest degree (LD), largest weighted degree (LWD), Largest enrolment degree (LE), and saturation degree (SD) [20], [21]. The heuristics are described as follows:

- Largest degree (LD): This technique orders the exams based on the largest number of conflicting examinations.
- Largest weighted degree (LWD): This heuristic is similar to the largest degree except the exams are ordered based on the number of students in conflict.
- Largest enrolment (LE): The exams are ordered based on the number of registered students in the exams.
- Saturation degree (SD): The exams are ordered based on the number of remaining timeslots available; exams with the least number of available timeslots in the timetable are given priority to be scheduled first. SD is a dynamic heuristic where the ordering of exams is updated as the exams being scheduled.

B. Local Search Approaches

Graph heuristics can generate an initial solution that is not optimum enough to consider a quality timetable. Hence local search meta-heuristics are frequently used to reduce the soft constraint violations as much as possible to get a quality solution from the initial solution. In this paper, three meta-heuristics are presented as they have been used for the proposed systems.

1) Late acceptance hill-climbing: Burke and Bykov [22] proposed late acceptance hill-climbing(LAHC) for escaping local optima produced by a greedy hill-climbing approach. In greedy hill-climbing, the candidate solution is compared with the immediate current one, but LAHC uses a delay comparison

mechanism where the candidate solution is compared with the solution of several iterations earlier. The algorithm starts with an initial feasible solution, and a new candidate solution is checked for acceptance in each iteration. A list of a specific length is used for memorizing the previous values of the current cost function used for acceptance criteria. Each time the candidate solution is compared with the last value of the list, and if better, it is accepted. When the acceptance procedure activates, the new cost is added at the beginning of the list, and the last element is deleted. The procedure is performed base on $v = ImodL$ formula, where L is the length of the frame, I is the i^{th} iteration, and v is the position.

2) Simulated annealing: 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 [23]. SA starts with a randomly generated initial solution, and in each iteration, it tries to improve the solution quality. If the neighboring solution is better than or equal to the current solution, it is replaced with the current one. Otherwise, acceptance of neighboring solution is decided on a probability function $exp(-\frac{f(s^*)-f(s)}{T})$, where $f(s^*)$ is a neighboring solution, $f(s)$ is the current solution, and T is a parameter known as temperature. Initially, the algorithm starts with a high T and periodically decreases the value using a cooling schedule until the temperature is zero or any terminal condition.

3) Great deluge algorithm: Great deluge (GDA) algorithm was proposed by Dueck [24]. The inspiration of this algorithm originated from the behavior in which 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 solutions. SA uses a probabilistic function for accepting the worst solutions, whereas GDA uses a more deterministic approach for that purpose. It is also found that GDA depends less on parameter tuning compared to SA. The only parameter in the GDA algorithm is the 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.

IV. System Architecture and Software Component

Fig. 1 depicts the overall system architecture of the examination timetabling scheduler. It consists of four different components: planning module, scheduling module, reporting module, and user interface.

A. Planning Module

Planning module deals with all the input data required for generating the solution. Here following steps are performed.

1) Constraints manager: Constraints manager handles all the hard and soft constraints associated with exam timetabling. Constraints can be modified according to user choice. Besides, another important function handled by this module is to compute the penalty cost of an exam solution. Here the objective function is employed for generating the penalty value of a given solution. This objective function can be predefined or it might be varied from one problem to another.

2) Exam conflict matrix: After all the necessary data are loaded into the system and constraints are defined, analysis of exam confliction is performed using a data structure named conflict matrix. The examination conflict matrix is a square matrix of dimension equal to the examination number. Each entry of the matrix indicates the number of students conflicting between the two examinations. Entry value of zero indicates no conflict between two exams, whereas a positive number means the existence of at least one conflict. This matrix facilitates managing different hard and soft constraints associated with timetabling and tracking the number of student enrolments in any pair of examinations.

B. Scheduling Module

This module is the central part of the overall system, aiming to generate a complete exam timetabling. Previous module passes necessary inputs so that scheduling module can produce appropriate exam timetabling based on user requirements. The scheduling of exams involves two steps. The first step is an initial feasible solution generation using an SD graph heuristic algorithm. In the second step, the quality of the solution is improved using local search algorithms. These include LAHC, SA, and GDA algorithms. Although not guaranteeing optimal solutions, they usually able to produce near-optimal solutions. Users can choose either of the steps for generating timetables and select desire algorithms with the appropriate parameters before execution. Note that improvement step does not execute independently, and it requires the solution vector of the first step. For example, improvement with the GDA algorithm phase is activated to produce a timetable when parameters such as decay rate and the number of iterations are properly defined, and initial feasible solutions produced by the graph heuristic algorithm are provided.

C. Reporting Module

Reporting module determines whether scheduling module is successfully generating the timetable or not according to the configuration assigned by users. Users can monitor the progress of the scheduling process, current penalty cost, execution time, and warning messages (if any) when scheduling module starts executing a timetable procedure. Once scheduling module produces a timetable, reporting module represents the final solution in a tabular form for the comprehension of everyone. Besides, subsidiary information such as total penalty costs (i.e., solution quality), execution times are also presented. At the end of the scheduling process, the document generator can be evoked to store final exam solutions (as an excel spreadsheet or pdf file format) in a disk for further uses.

D. User Interface

There is a graphical user interface that works at the top level. Users (e.g., students, teachers, human scheduler) can

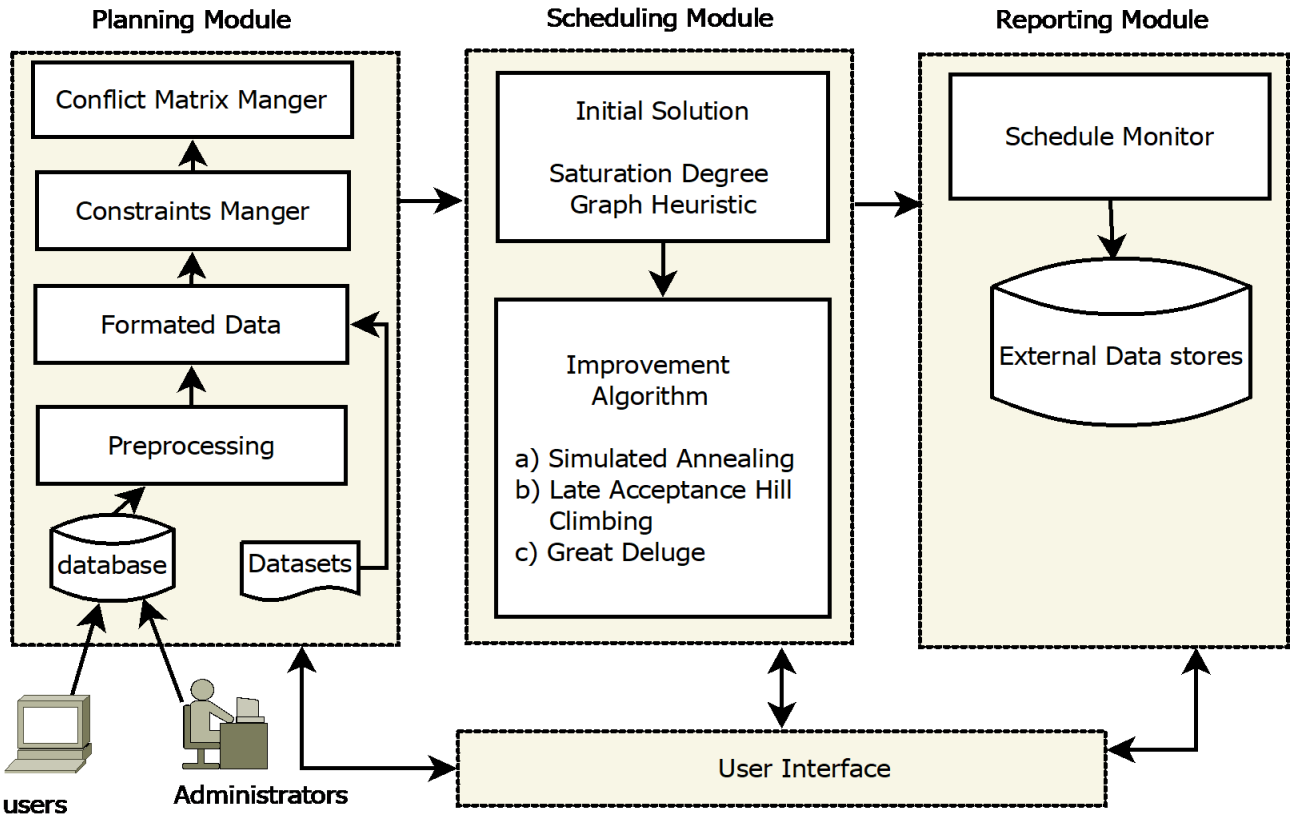


Fig. 1: Overview of system architecture

interact with all the modules using this interface. For example, visualizing the data, controlling the settings, and flow of the process can be done efficiently using this interface. Note that the whole system is developed using Java SE 1.7, and its swing library is deployed for GUI implementation.

1) Description of the system interface: The interface is designed in a straightforward way so that a user can operate the system with minimal effort and get better user experience. In addition to the graphical user interface, a command-line interface is provided for advanced users. As it is hard to illustrate all the options of the user interface, some selected screenshot of the GUI of automated exam timetabling is presented in Fig. 2, Fig. 3, and Fig. 4. The system window contains different key components, including a menu bar, toolbar, and tab panels. The menu bar at the top is used to load the dataset. Below the menu bar is a toolbar which provides important options such as construction and improvement phase of timetable. Different tab panels are associated with each tool, with each tab being used for performing different actions. For instance, the Data file tab of the construction tool contains data sets used for scheduling. Here users can customize a dataset, impose different constraints by six command buttons, and update other relevant information (see Fig. 2). A sample execution of an exam timetable process is shown in Fig. 3. There are two different types of components on the GUI that are used to monitor the status of the execution process. The text area shows the successful allocation of exams into the time slot and rooms (quality of the solution), and the progress bar indicates the percentage of progression of an exam timetable. The left side of the window contains some input fields (drop-down combo boxes, text boxes, etc.), which are used to select a search algorithm, tune parameters, and set stopping criterion. For example, for a sample execution with a GDA algorithm, users have to set the decay rate and the number of iterations. One may change the configuration and even run multiple executions simultaneously, as every execution of the scheduling process is an independent thread. Fig. 4 shows the presentation of the final examination timetable results in the data grid. Some performance measures, such as penalty cost (i.e., quality of timetabling) and the execution time are also displayed on the left side of data grid, and users can save and retrieve exam schedules for further uses.

V. Results and Discussion

Whether the timetabling system is viable in solving examination timetabling correctly, it has been tested with eight instances of ITC2007 exam dataset. In the experiment, graph heuristic SD is used for producing the initial solution, and three local search meta-heuristics are used individually for optimizing the initial solutions in the improvement phase. Three variations of the neighbourhood structures have been employed within an improvement algorithm. Their explanation is outlined as follows:

- $N1$: An examination is selected randomly and moves it to a random time slot.
- $N2$: Two examinations are selected randomly and swapping is occurred between their time slots.

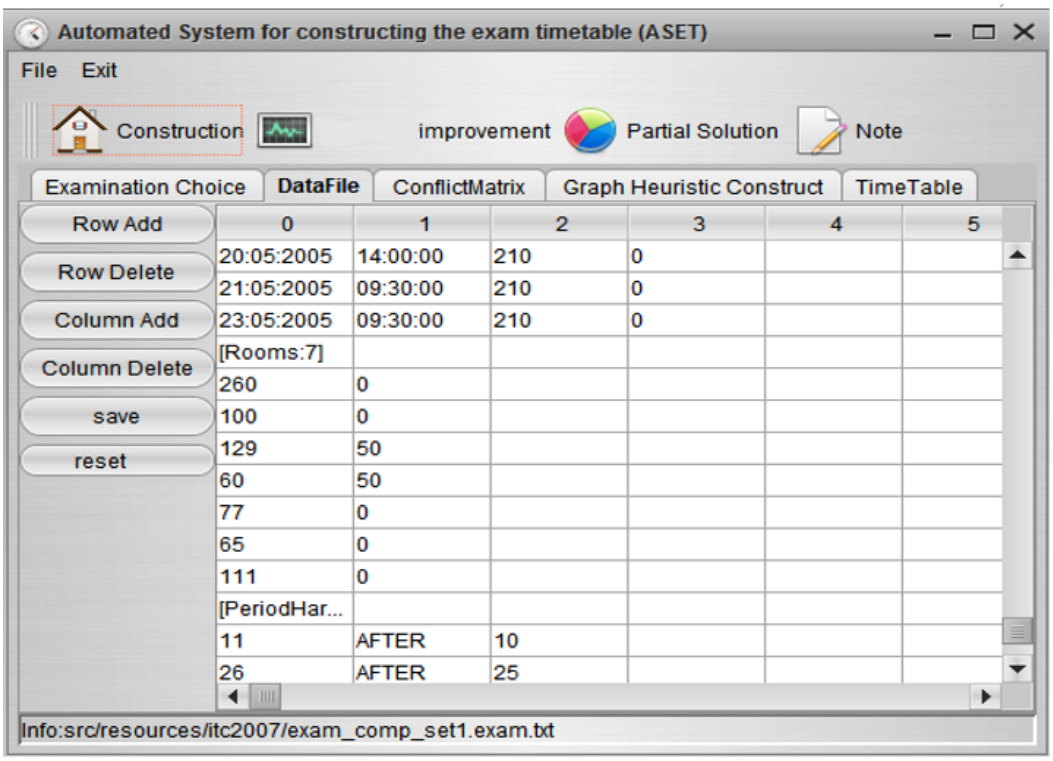


Fig. 2: Input with various entities

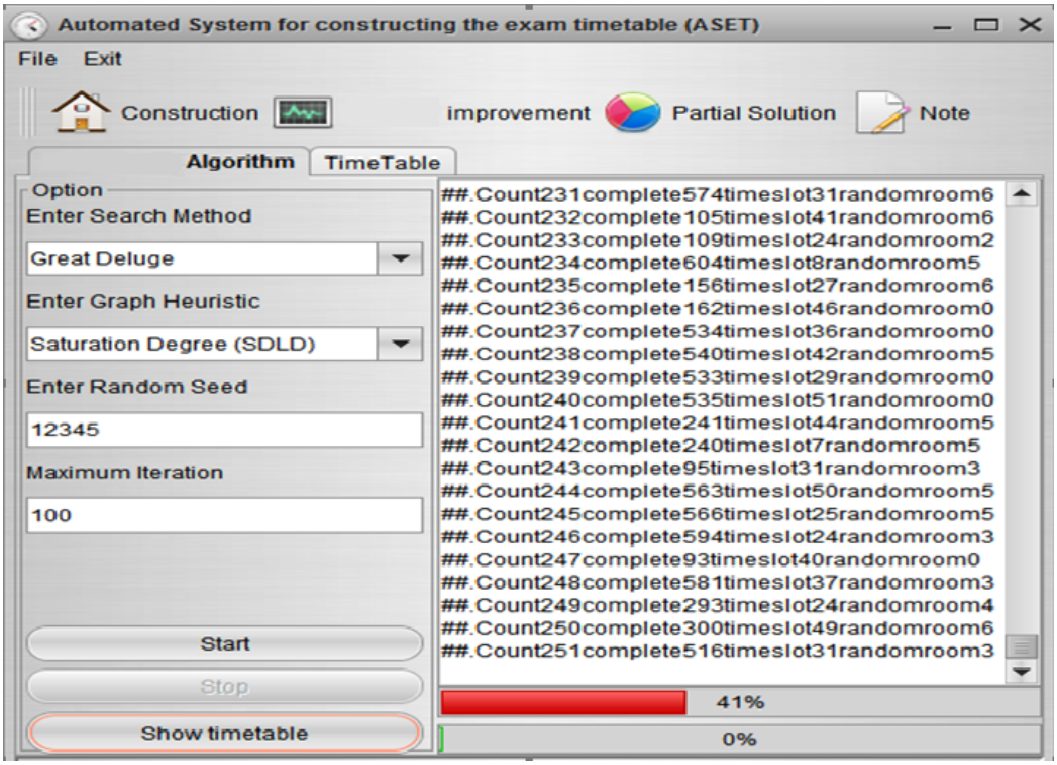
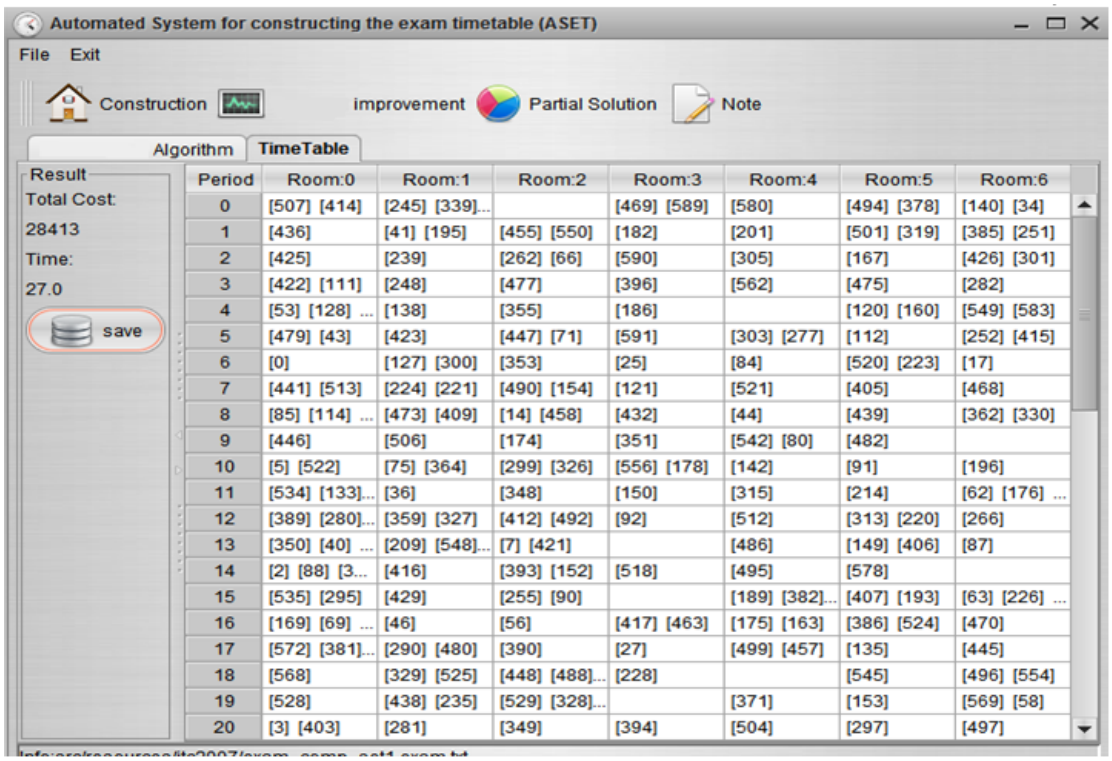


Fig. 3: Executing of a timetabling with given parameters



Period	Room:0	Room:1	Room:2	Room:3	Room:4	Room:5	Room:6
0	[507] [414]	[245] [339]...		[469] [589]	[580]	[494] [378]	[140] [34]
1	[436]	[41] [195]	[455] [550]	[182]	[201]	[501] [319]	[385] [251]
2	[425]	[239]	[262] [66]	[590]	[305]	[167]	[426] [301]
3	[422] [111]	[248]	[477]	[396]	[562]	[475]	[282]
4	[53] [128] ...	[138]	[355]	[186]		[120] [160]	[549] [583]
5	[479] [43]	[423]	[447] [71]	[591]	[303] [277]	[112]	[252] [415]
6	[0]	[127] [300]	[353]	[25]	[84]	[520] [223]	[17]
7	[441] [513]	[224] [221]	[490] [154]	[121]	[521]	[405]	[468]
8	[85] [114] ...	[473] [409]	[14] [458]	[432]	[44]	[439]	[362] [330]
9	[446]	[506]	[174]	[351]	[542] [80]	[482]	
10	[5] [522]	[75] [364]	[299] [326]	[556] [178]	[142]	[91]	[196]
11	[534] [133]...	[36]	[348]	[150]	[315]	[214]	[62] [176] ...
12	[389] [280]...	[359] [327]	[412] [492]	[92]	[512]	[313] [220]	[266]
13	[350] [40] ...	[209] [548]...	[7] [421]		[486]	[149] [406]	[87]
14	[2] [88] [3...	[416]	[393] [152]	[518]	[495]	[578]	
15	[535] [295]	[429]	[255] [90]		[189] [382]...	[407] [193]	[63] [226] ...
16	[169] [69] ...	[46]	[56]	[417] [463]	[175] [163]	[386] [524]	[470]
17	[572] [381]...	[290] [480]	[390]	[27]	[499] [457]	[135]	[445]
18	[568]	[329] [525]	[448] [488]...	[228]		[545]	[496] [554]
19	[528]	[438] [235]	[529] [328]...		[371]	[153]	[569] [58]
20	[3] [403]	[281]	[349]	[394]	[504]	[297]	[497]

Fig. 4: Desired exam timetable

- $N3$: Two time slots are selected randomly and all examinations between the two time slots are swapped.

The termination criteria for the improvement phase are fixed at 10000 iterations. Besides, 30 individual run is also performed for each instance. The following parameters are set for the local searches that are shown in Table III. Note that other setting of parameters could have been selected, but these parameter values have been selected according to the values used in the scientific literature.

TABLE III: Parameter Setting for experiments

Algorithm	Parameter	Value(s)
SA	Cooling rate	0.1
	Temperature	5000
LAHC	Frame size	500
GDA	Decay rate	0.1

From the experimental results in Table IV, it can be deduced that the proposed system can solve the ITC2007 exam benchmark dataset effectively. For all of the instances of the dataset, the best and the average values are highlighted after 30 individual runs. Graph heuristic SD produces feasible solutions for all of the instances, and local search approaches further improve the quality of the solutions. Among these three local search algorithms, 5 out of 8 cases (e.g., Exam_1, Exam_3, Exam_5, Exam_6, and Exam_7) LAHC performed the best results followed by GDA with instances Exam_2 and Exam_8, and SA with Exam_4. It is apparent that graph heuristic does not produce quality results because it considers only hard constraints. On the other hand, local search optimization algorithms produce better results compared to graph heuristics because the local search can improve the solution by reducing the soft constraint violations. Note that this paper does not aim to find the performance of the algorithms that suite the best. Instead, it shows the capabilities of graph heuristic and some local search algorithms for solving the exam timetable interactively. It is up to users to decide what will be the most useful algorithm in their particular circumstances. The advantages of the proposed interactive system are highlighted below:

- As the system has been developed using Java, it can run on a computer running both Windows and Linux (i.e., platform independence). Moreover, the incorporation of multi-thread assists the user to execute and analyze more than one exam instance simultaneously.
- This interactive tool is able to construct a complete timetable within a short time compared to the human scheduler, which usually takes long preparation in advance. It is also notable that maintaining hard constraints strictly and minimization of soft constraint violations for a large number of exams are challenging for the human scheduler. In contrast, an automated scheduler can perform the tasks firmly and effectively.
- This tool is useful as it can efficiently utilize institutional resources (i.e., room and timeslot utilization) as well as fulfill the major requirements requested by the

TABLE IV: Performance (penalty values) comparison between SD, LAHC, SA and GDA on ITC2007 exam dataset

Instances	SD		LAHC		SA		GDA	
Measure	Best	Avg	Best	Avg	Best	Avg	Best	Avg
Exam_1	25,989	26,769.45	12,421	13,048.88	12,537	14,042.37	12,483	13,858.68
Exam_2	30,960	32,135.67	2,807	3,766.79	2,911	3,553.26	2,789	3,578.79
Exam_3	85,356	88,374.43	43,098	47,160.20	44,173	48,060.62	43,241	47,560.63
Exam_4	41,702	42,323.38	34,241	34,937.07	34,152	34,834.18	34,417	34,744.46
Exam_5	132,953	133,873.50	15,643	16,773.46	15,816	16,891.51	15,690	16,612.17
Exam_6	44,160	48,729.67	29,630	33,880.08	30,116	34,910.16	29,845	34,150.38
Exam_7	53,405	56,366.08	19,080	21,612.42	20,071	21,518.31	19,178	21,821.75
Exam_8	92,767	96,465.32	23,315	25,002.29	23,411	25,801.79	22,891	25,152.65

students and invigilators. Institutional personnel can easily use and maintain the software without having prior programming knowledge.

- Flexibility in changing of input settings (e.g., constraints, exams, resources), support of interactive parameters tuning, and selection of different execution methods (either initial feasible solution with graph heuristic or initial solution followed by a near-optimum solution using local searches) are some salient features of the system, which makes it a robust interactive tool. Users can observe the effects of the different configurations on the output quality of the timetable.

- Although the system includes predefined eight instances of ITC2007 dataset, a provision has been kept for the users to modify or add new user-defined exam instances.

VI. Conclusions

This paper aims to generate an easy to use interactive examination timetabling software whereby graph heuristics and different local search algorithms are employed as solution methods. SD graph heuristic generates an initial feasible solution, whereas local search algorithms such as SA, GDA, and LAHC work as an optimizer for producing near-optimum solutions for the exam dataset. This proposed system is developed using Java and tested successfully on real-world dataset named ITC2007 exam dataset. The software is flexible and robust that outweighs the manual approaches. Users can automatically produce dataset from student registrations, select preferred hard and soft constraints, employ different improvement algorithms with desirable parameters and eventually produce a quality timetable within a reasonable time frame. The system could be scaled up by including different population-based search algorithms, which could provide more efficiency in the improvement phase. The usability of the software can also be enhanced to attain a satisfactory user experience.

Acknowledgment

This research is supported by Ministry of Posts, Telecommunication and Information Technology, Government of People's Republic of Bangladesh (Memo no: 56.00.0000.28.33.042.15-509)

References

[1] J. Johnes, "Operational research in education," *European Journal of Operational Research*, vol. 243, no. 3, pp. 683 – 696, 2015.

[2] P. Boizumault, Y. Delon, and L. Peridy, "Constraint logic programming for examination timetabling," *The Journal of Logic Programming*, vol. 26, no. 2, pp. 217 – 233, 1996.

[3] A. Cataldo, J.-C. Ferrer, J. Miranda, P. A. Rey, and A. Sauré, "An integer programming approach to curriculum-based examination timetabling," *Annals of Operations Research*, vol. 258, no. 2, pp. 369–393, 2017.

[4] N. R. Sabar, M. Ayob, R. Qu, and G. Kendall, "A graph coloring constructive hyper-heuristic for examination timetabling problems," *Applied Intelligence*, vol. 37, no. 1, pp. 1–11, 2012.

[5] M. Mohmad Kahar and G. Kendall, "A great deluge algorithm for a real-world examination timetabling problem," *Journal of the Operational Research Society*, vol. 66, no. 1, pp. 116–133, 2015.

[6] Y. Bykov and S. Petrovic, "A step counting hill climbing algorithm applied to university examination timetabling," *Journal of Scheduling*, vol. 19, no. 4, pp. 479–492, 2016.

[7] P. Amaral and T. C. Pais, "Compromise ratio with weighting functions in a tabu search multi-criteria approach to examination timetabling," *Computers & Operations Research*, vol. 72, pp. 160–174, 2016.

[8] M. Battistutta, A. Schaerf, and T. Urli, "Feature-based tuning of single-stage simulated annealing for examination timetabling," *Annals of Operations Research*, vol. 252, no. 2, pp. 239–254, 2017.

[9] N. Pillay and W. Banzhaf, "An informed genetic algorithm for the examination timetabling problem," *Applied Soft Computing*, vol. 10, no. 2, pp. 457–467, 2010.

[10] O. Abayomi-Alli, A. Abayomi-Alli, S. Misra, R. Damasevicius, and R. Maskeliunas, "Automatic examination timetable scheduling using particle swarm optimization and local search algorithm," in *Data, Engineering and Applications*, pp. 119–130, Springer, 2019.

[11] 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, no. 1, pp. 37–54, 2015.

[12] H. Babaei, J. Karimpour, and A. Hadidi, "A survey of approaches for university course timetabling problem," *Computers & Industrial Engineering*, vol. 86, pp. 43–59, 2015.

[13] R. Qu, E. K. Burke, B. McCollum, L. T. Merlot, and S. Y. Lee, "A survey of search methodologies and automated system development for examination timetabling," *Journal of scheduling*, vol. 12, no. 1, pp. 55–89, 2009.

[14] S. Piechowiak and C. Kolski, "Towards a generic object oriented decision support system for university timetabling: an interactive approach," *International Journal of Information Technology & Decision Making*, vol. 3, no. 01, pp. 179–208, 2004.

[15] J. J. Thomas, A. T. Khader, B. Belaton, and E. Christy, "Visual interface tools to solve real-world examination timetabling problem," in *2010 Seventh International Conference on Computer Graphics, Imaging and Visualization*, pp. 167–172, IEEE, 2010.

[16] M. Ayob, A. R. Hamdan, S. Abdullah, Z. Othman, M. Z. A. Nazri, K. A. Razak, R. Tan, N. Baharom, H. A. Ghafar, R. M. Dali, *et al.*, "Intelligent examination timetabling software," *Procedia-Social and Behavioral Sciences*, vol. 18, pp. 600–608, 2011.

[17] Z. Chunbao and T. Nu, "An intelligent, interactive & efficient exam scheduling system (iieess v1. 0)," *Proceeding of the Practice and Theory of Automated Timetabling (PATAT), Norway*, pp. 437–450, 2012.

[18] I. Ober, "A variant of the high-school timetabling problem and a software solution for it based on integer linear programming," 2016.

[19] T. Müller, "Itc2007 solver description: a hybrid approach," *Annals of Operations Research*, vol. 172, no. 1, p. 429, 2009.

[20] 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, no. 1, pp. 291–315, 2012.

[21] A. K. Mandal and 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)*, pp. 530–534, IEEE, 2015.

[22] E. K. Burke and Y. Bykov, "A late acceptance strategy in hill-climbing for exam timetabling problems," in *PATAT 2008 Conference, Montreal, Canada*, pp. 1–7, 2008.

[23] K. Bouleimen and H. Lecocq, "A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version," *European journal of operational research*, vol. 149, no. 2, pp. 268–281, 2003.

[24] G. Dueck, "New optimization heuristics: The great deluge algorithm and the record-to-record travel," *Journal of Computational physics*, vol. 104, no. 1, pp. 86–92, 1993.