A Practical Three-Phase ILP Approach for Solving the Examination Timetabling Problem

Feras Al-Hawari, Mahmoud Al-Ashi, Fares Abawi, and Sahel Alouneh
Computer Engineering Department, German Jordanian University, Amman, Jordan
{firas.alhawari, m.alashi, f.abawi, sahel.alouneh}@gju.edu.jo

Abstract: A practical mathematical programming based approach is introduced for solving the examination timetabling problem at the German Jordanian University (GJU), whereby the complex process of acquiring a feasible examination timetable is simplified by subdividing it into three smaller sub-problems (phases). Accordingly, the exams are initially allocated to time slots in phase one, the time slots are then allotted to days in phase two, and finally in phase three the exams are assigned to rooms based on the number of students taking each exam and capacities of the rooms. The solution for each phase is acquired based on an integer linear programming formulation, while satisfying a set of hard constraints that ensure comfortable exam timetables for all students and meet the desired requirements set by GJU administrative staff. Furthermore, the solver can be controlled and launched from a student information system named MyGJU Admin, which enabled registrars at the university to easily, quickly, and accurately generate final exam timetables in several standard formats. Moreover, the approach was validated based on recent GJU registration information as well as real-world benchmark data.

Keywords: Examination Timetabling, Optimization, Scheduling, Integer Linear Programming, Graph Coloring.

INTRODUCTION

Exam timetabling is a very important and time critical task that faces registrars every semester in most educational institutions. This task is very complex and requires assigning a date, time, and room to every exam while ensuring that time (e.g., limited number of days, certain time intervals, fixed working hours), spatial (e.g., availability and capacity of rooms), and other (e.g., relaxing, and preventing conflicts in, the schedules of the students) constraints are satisfied. With the increasing number of students, exams, and demands, solving the Exam Timetabling Problem (ETP) manually is not a practical option and hence providing a computational solution for it attracted the attention of many researchers from the 1960s as in [1] till recently as in [2].

The ETP is known to be as an NP-Complete optimization problem [3] (i.e., it is unlikely to solve it optimally in polynomial time). Therefore, researchers tried and are still experimenting with many different ways that are based on mathematical models, heuristic techniques, or a combination of different algorithms to find quick and acceptable solutions for it. For example, in the survey presented in [4], the exam timetabling research was classified based on the techniques used to solve the problem such as: graph based [5, 6], local search based [7, 8], population based [9, 10], and other methods. In summary, the proposed solutions compare to each other based on the used techniques, the hybridization of different methods, the decomposition of the problem into smaller sub-problems, and the hard and soft constraints taken into consideration.

In this paper, a new technique is introduced to solve the ETP with emphasis on issues related to GJU and its set of rules and limiting constraints. The proposed method has three novelties. First, a feasible (i.e., not necessarily optimal) solution to the problem is found by segmenting it into three phases, which reduces the number of constraints to be considered in each phase and hence allows the optimizer to reach a desired solution in a quick manner with reduced memory demands. Noting that, an Integer Linear Programming (ILP) [11] based approach is used to find the solution for each phase. Second, a comprehensive set of hard constraints is considered to generate an exams schedule that is comfortable to all students (e.g., a student should not have more than two exams on the same day) and meets the needs of the different faculties at GJU (e.g., fixing the date of an exam, conducting an exam in a specific set of rooms). Third, the same exam can be allocated to one or multiple rooms unlike most of the similar techniques.

Similarly to the method in this paper, several other papers such as those in [12-17] discussed integer programming based techniques to solve the ETP. However, to the best of our knowledge, most of those methods tried to solve the problem in one phase, which usually results in a system of equations with a large number of variables and hence solving it is very CPU and memory intensive. Unlike [12], the hard constraint to

prevent a student from having two or more exams in the same day as in this paper was not considered in [13-16]. In [17], a hybrid adaptive decomposition approach was used to break the exams into difficult and easy sets before using an ILP approach to obtain a solution.

Some of the approaches that did not use a mathematical ILP model (i.e., unlike the approach in this paper) to solve the ETP will be briefly discussed next. Such approaches either used graph coloring, metaheuristic, hybrid, or other methods to find a solution for the ETP. They can also be categorized into two groups based on whether all the hard constraints in this paper were considered or not.

The first group of papers (i.e., the group that considered the set of hard constraints as in this paper) is discussed first. In [2], two column generation algorithms were used to solve the ETP. Whereas in [18], hyper heuristic approaches were utilized. In [19], an adaptive linear combination of heuristics with a heuristic modifier under the framework of adaptive strategies was proposed. In [20], a search algorithm that consists of several phases is introduced. In the first (construction) phase, a complete solution is found using an iterative forward search algorithm. In the later phases, a local optimum is found using a combination of a hill climbing algorithm and great deluge technique. The method in [21] hybridized bin packing heuristics to assign exams to time slots and rooms. In [22], the solution is based on graph coloring heuristics that were hybridized to generate four new low level heuristics. In [23], the ETP specified in [24] was solved using a variable neighborhood search methodology. In [25], a random iterative graph based hyper-heuristic was used to produce a collection of heuristic sequences to construct solutions of different quality.

Next, the second group of papers (i.e., that did not consider all hard constraints used in this paper) is presented. For example, a hybrid bee colony optimization approach was used in [26]. In [27], a sequential graph coloring with the largest enrolment first heuristic was used to construct a conflict-free examination timetable and then a simulated annealing heuristic was used to fit examinations into rooms, while satisfying the back-to-back constraint (i.e., spreading the examinations evenly to give the students enough revision time between examinations). In [28, 29], hill climbing and great deluge local search were used to solve the problem. In [30], a hybrid harmony search algorithm was used. In [31], decomposition as well as a graph coloring heuristic were used. In [32], a hyper heuristic approach was used. In [33], heuristics and a stochastic algorithm called the roulette wheel graph coloring were used to solve the problem. In that the algorithm was also tested on the examination timetabling benchmark datasets in [34]. A graph-coloring-based method was utilized in [6], but without considering the actual distribution of exam sessions to rooms. In [35], the solution of the course-timetabling problem was used to construct an initial solution to the examination timetable. Finally, in [5] a hybrid two phase method was introduced to tackle the ETP.

The rest of the paper is organized as follows. In the Method Overview section, the hard constraints under consideration and an overview of the proposed method are provided. In the Method Implementation section, the adopted nomenclature, problem setup, and decomposing the ETP into three sub-problems are discussed. In the AMPL Format Representation section, a simple example is used to show how the problem formulation is represented in AMPL format. In the Validation and Results section, the method is validated based on GJU registration information (using the MyGJU Admin tool [36, 37]) and real-world benchmark data. Finally in the Conclusions and Future Work section, conclusions and future work are presented.

METHOD OVERVIEW

In this section, the hard constraints under consideration in the ETP problem formulation are defined. Moreover, the decomposition of the ETP into three sub-problems to be able to find a feasible solution for it is discussed.

Hard Constraints

By definition a hard constraint must be satisfied by the solver to obtain a feasible solution i.e., it cannot be violated by the solver. The following hard constraints were considered in this paper:

i. A student cannot have more than one exam simultaneously
ii. All exams must be scheduled
iii. An exam must take place within the available time slots and dates
iv. The number of exams to be held simultaneously is limited due to the limitation on room availability
v. A student cannot have more than two (or more) exams on any given day
vi. The maximum number of time slots per day should be respected

vii. A room cannot be occupied by two exams at the same time
viii. An exam can be held in multiple rooms up to a specified limit
ix. The room capacity should not be exceeded
x. Some exams can be held only on specific days during the examination period
xi. Some exams can be allocated only to specific rooms

Problem Decomposition

Solving the ETP in one phase can be out-of-reach [12] as the problem formulation may contain a large number of variables and hence results in out-of-memory issues. Therefore, in this paper the ETP is decomposed into three sub problems and then practically solved in three phases. Accordingly, a later phase depends on the results of its preceding phase(s). A graph coloring based ILP formulation is used to find a solution in phases one and two. Solving a graph coloring problem requires assigning colors to vertices such that no two adjacent (i.e., connected with an edge) vertices share the same color, while minimizing the number of colors used to cover all vertices.

Based on that, phase 1 is formulated to assign time slots (colors) to exams (vertices) such that no two adjacent exams will be assigned the same time slot (i.e., color). An exam is adjacent to another exam when they share a student i.e., the student cannot take both exams simultaneously. Therefore, the objective of this phase is to minimize the number of time slots used while preventing any student from having more than one exam simultaneously (i.e., satisfying hard constraint i). Note that, the maximum number of time slots is bounded by $|T|$, which should not exceed the total number of exams (i.e., vertices).

Phase 2 is required to assign days (colors) to the time slots (vertices) that were determined in phase 1, given that a student shall not have more than two (or more) exams in any given day i.e., a day shall not contain more than two time slots with exams having a common student in them.

Whereas, phase 3 is needed to assign the exams to the available rooms, while not exceeding the capacities of the rooms in comparison with the number of students taking the corresponding exams. The formulation in this phase is based on the time slots and days that were determined in phase 1 and phase 2, respectively.

METHOD IMPLEMENTATION

In this section, the sets and indices used in the problem formulation are given. Furthermore, the problem setup and ILP formulations for the three phases are discussed. Note that, in each phase the problem is translated into mathematical programming equations according to the AMPL (A Mathematical Programming Language) program format [39] in order to be solved by the CPLEX solver [40], while meeting the desired hard constraints.

Sets and Indices

The sets and indices that are used in the formulation of the three phases (sub problems) are defined as follows:

- $\boldsymbol{C}$ is the set of all exams
- c is the index of an exam i.e., $c \in \boldsymbol{C}$
- $\boldsymbol{E}$ is a set that contains students taking each exam $c \in \boldsymbol{C}$
- E_c is the number of students taking exam c
- $\boldsymbol{T}$ is the set of time slots
- t is the index of a time slot i.e., $t \in \boldsymbol{T}$
- $\boldsymbol{D}$ is the set of days
- d is the index of a day i.e., $d \in \boldsymbol{D}$
- $\boldsymbol{T_d}$ is the set of time slots (tuple) allocated to a day
- $\boldsymbol{R}$ is the set of rooms
- r is the index of a room i.e., $r \in \boldsymbol{R}$
- $\boldsymbol{P}$ is a set that contains the capacity of each room $r \in \boldsymbol{R}$
- P_r is the capacity of room r
- $\boldsymbol{S}$ is the set of all students enrolled in the semester under consideration
- s is the index of a student i.e., $s \in \boldsymbol{S}$
- C_s is the list of exams (tuple) that student s is taking
- $\boldsymbol{F}$ is the set of student exam lists (tuples)
- T_s is the list of time slots (tuple) in which the exams for student s can be allocated
- $\boldsymbol{J}$ is the set of student time slots lists (tuples)
- T_d is a predefined value for the maximum number of time slots per day
- C_t is a predefined value for the maximum number of exams per time slot
- $\boldsymbol{R_{acc_c}}$ is a set that contains all rooms that can be assigned to exam c
- $\boldsymbol{R_{rej\_c}}$ is a set that contains all rooms that cannot be assigned to exam c

Problem Setup

The proposed method is integrated in a web-based student information system named MyGJU Admin [36-38]. This system enables registrars at

GJU to perform final exams scheduling as well as many other tasks like rooms setup, course sections management, admission, student information management, registration, grades processing, and graduation. Therefore, all the data needed (e.g., exams, students, enrollments, rooms) to populate the aforementioned sets (e.g., $\boldsymbol{C}$, $\boldsymbol{R}$, $\boldsymbol{S}$, $\boldsymbol{F}$) that are used in the following formulations and to solve the ETP can be first retrieved from the corresponding MyGJU database tables, and then fed to the solver to obtain the desired solution.

Phase 1: Exams to Time Slots Assignment

The following formulation is used to guarantee that no exam in each student exams tuple (i.e., in each C_s) will be assigned to the same time slot (i.e., to satisfy hard constraint **i**):

$$\forall s \in \{1, \dots, |\boldsymbol{F}|\}, \forall t \in \{1, \dots, |\boldsymbol{T}|\}: \quad (\sum_{c=1}^{|C_s|} x_{c,t} \leq 1) \qquad (1)$$

where the binary variable $x_{i,j} = 1$ when exam i is assigned to time slot j.

In order to guarantee that every exam will be assigned to exactly one time slot (i.e., satisfy hard constraints **ii** and **iii**), the following constraint set is needed:

$$\forall c \in \{1, \dots, |\boldsymbol{C}|\}: \quad (\sum_{t=1}^{|T|} x_{c,t} = 1) \qquad (2)$$

Moreover, the number of exams in each time slot can be guaranteed not to exceed a pre-defined value C_t (i.e., satisfy hard constraint **iv**), usually set by registrars, as shown below:

$$\forall t \in \{1, \dots, |\boldsymbol{T}|\}: \quad (\sum_{c=1}^{|C|} x_{c,t} \leq C_t) \qquad (3)$$

Furthermore, to satisfy hard constraint **x**, an additional step is introduced as follows:

$$\forall c_i, c_j \in \boldsymbol{C}, d_n, d_m \in \boldsymbol{D}: (c_i \rightarrow d_n \cap c_j \rightarrow d_m) \rightarrow x_{i,t} + x_{j,t} \leq 1 : \forall t \in \boldsymbol{T}, \mathrm{i} \neq \mathrm{j} \qquad (4)$$

The previous constraint denotes that, if exam c_i is to be held in day d_n, and exam c_j is to be held in day d_m, then c_i and c_j should not occur in the same time slot.

Finally, if a binary variable u_t is equal to one when there is at least one course c assigned to time slot t as guaranteed by:

$$\forall t \in \{1, \dots, |\boldsymbol{T}|\}: \quad (\sum_{c=1}^{|C|} x_{c,t} - u_t \leq 0) \qquad (5)$$

Then, the number of used time slots can be minimized based on the following objective:

$$\text{minimize: } \sum_{t=1}^{|T|} u_t \qquad (6)$$

Phase 2: Time Slots to Days Assignment

As a preliminary step in this phase, the student exams tuples in set $\boldsymbol{F}$, that are used in phase 1, are first processed to generate set $\boldsymbol{J}$ by replacing the exams in each tuple with the corresponding time slots that they were associated with in phase 1. Henceforth, if exam i was assigned to time slot j in the previous phase, then each occurrence of i in any student exams tuple C_s will be replaced by j to obtain the student time slots tuple T_s.

Assuming that a binary variable h_s is equal to one only when a student whose tuple T_s has n exams in day d. Hence, a student will not have more than n exams on any given day d (i.e., satisfy constraint **v**) when meeting the following:

$$\forall s \in \{1, \dots, |\boldsymbol{J}|\}, \forall d \in \{1, \dots, |\boldsymbol{D}|\}: \quad (\sum_{t=1}^{|T_s|} y_{t,d} - h_s \leq (n-1)) \qquad (7)$$

where $y_{i,j} = 1$ when time slot i in T_s is assigned to day j. Also, based on equation 7, if a student had n exams in one day, the value of his/her corresponding h_s will be automatically set to one. Noting that in the case of GJU n is equal to two.

The following equation is further used to minimize the number of students having n exams per day:

$$\text{minimize: } \sum_{s=1}^{|S|} h_s \qquad (8)$$

In addition, to guarantee that each time slot will be assigned to one and only one day, the following constraints need to be met:

$$\forall t \in \{1, \dots, |\boldsymbol{T}|\}: \quad (\sum_{d=1}^{|D|} y_{t,d} = 1) \qquad (9)$$

Where $y_{i,j} = 1$ when time slot i is assigned to day j.

Moreover, the number of time slots per day can be limited to a pre-defined value T_d (i.e., satisfy constraint **vi**), usually set by registrars, as follows:

$$\forall d \in \{1, \dots, |\boldsymbol{D}|\}: \quad (\sum_{t=1}^{|T|} y_{d,t} \leq T_d) \qquad (10)$$

To further satisfy hard constraint **x**, the time slots that were assigned to certain exams previously must be linked to certain days according to the formula below:

$$(x_{c,t_i} = 1 \cap c \rightarrow d_j) \rightarrow y_{t_i,d_j} = 1 \qquad (11)$$

The previous equation denotes that if exam c is to be held in day d_j and c was assigned to time slot t_i, then t_i must be a time slot in day d_j.

Phase 3: Exams to Rooms Assignment

The goal of this phase is to assign exams to rooms. Hence, at the end of this phase each exam should be assigned to room(s) at the time-slot and day that were determined in the previous two phases, while meeting all the desired hard constraints.

Assuming that u_c is a binary variable with a value equals to 1 when exam c is assigned to n rooms. Then, hard constraints **viii** and **ix** can be met as follows:

$$\forall c \in \{1, \ldots, |\boldsymbol{C}|\}: \quad (\sum_{r=1}^{|R|} x_{c,r} * P_r \geq E_c) \quad (12)$$

It is also worth noting that, during the examination period, a room that fits x students on normal days is assigned only $x/2$ students. This step is performed to avoid overcrowding the rooms and provide students with a comfortable examination environment.

To ensure that each exam will occur in exactly one room (i.e., meet hard constraint **vii**), the following equation needs to be satisfied:

$$\forall c \in \{1, \ldots, |\boldsymbol{C}|\}: \quad (\sum_{r=1}^{|R|} x_{r,c} = 1) \quad (13)$$

where $x_{\mathrm{i,j}} = 1$ when room i is to host exam j.

Moreover, to satisfy hard constraint **xi** an additional equation must be added and one of two cases may occur:

i. If a room is **not** to be allowed to hold a certain exam, then:

$$\forall r \in \boldsymbol{R} \cap r \in \boldsymbol{R_{rej\_c}}: \quad x_{c,r} = 0 \quad (14)$$

where $\boldsymbol{R_{rej\_c}}$ is the set that contains all rooms that are not to be assigned to exam c.

ii. If an exam is **only** to be held in certain rooms, then:

$$\forall r \in \boldsymbol{R} \cap r \notin \boldsymbol{R_{acc\_c}}: \quad x_{c,r} = 0 \quad (15)$$

where $\boldsymbol{R_{acc\_c}}$ is the set that contains all rooms that can be assigned to exam c.

AMPL FORMAT REPRESENTATION

A small exam timetabling problem is considered in this section to illustrate how the variables and equations used to formulate the sub-problems for phase 1 and phase 2 can be represented in AMPL format (note that the phase 3 representation is not shown here as it can be done in a similar manner). For example, assuming that a user would like to schedule the four exams shown in Table 1 over a two days period. Given that, each day contains three time slots (i.e., a total of 6 time slots in 2 days), the exams taken by each student are as shown in Table 2, and no student should have more than two exams on any given day.

The phase 1 formulation can be represented in AMPL syntax as shown in Figure 1. At line 1, CPLEX is set as a solver. Whereas, sets $\boldsymbol{T}$, $\boldsymbol{C}$, and $\boldsymbol{F}$ as well as variables x and u_t are declared at lines 2, 3, 4, 5, and 6, respectively. Sets $\boldsymbol{T}$, $\boldsymbol{C}$, and $\boldsymbol{F}$ are initialized at lines 8, 9, and 10, respectively. Furthermore, equations 1, 2, 3, 5, and 6 are defined at lines 11, 12, 13, 14, and 15, respectively.

The phase 2 formulation can be represented in AMPL syntax as shown in Figure 2. At line 2, the CPLEX options the number of solutions and time limit are set to 2 solutions and 120s, respectively. Whereas, sets $\boldsymbol{D}$, $\boldsymbol{T}$, and $\boldsymbol{J}$ as well as variables x and has_2 (i.e., h_s) are declared at lines 3, 4, 5, 6, and 7, respectively. Also, sets $\boldsymbol{D}$, $\boldsymbol{T}$, and $\boldsymbol{J}$ (given that only time slots 1-4 were needed to solve phase 1) are initialized at lines 9, 10, and 11, respectively. Moreover, equations 7, 8, 9, and 10 are defined at lines 12, 13, 14, and 15, respectively. Also, the solver is called at line 16.

VALIDATION AND RESULTS

The proposed method was validated using the GJU first 2016/2017 semester registration information as well as the University of Toronto benchmark data [41] as will be discussed in the following two subsections.

Note that all of the experiments discussed next were executed on a machine with an Intel Core i7-4710MQ CPU running at 2.5GHz, with 16GB RAM, and on which a windows 10 enterprise operating system is installed. Also, CPLEX version 12.6.3.0 was used in all cases.

Table 1: Students enrolled in each exam

Exam	Student Id	Count
CRS1	1, 2, 3, 4	4
CRS2	1,3,4,5,7,9	6
CRS3	3,4,5,7,8,10	6
CRS4	3,5,6,10	4

Table 2: Exams of each student

Student Id	Exams
1	CRS2, CRS1
2	CRS1
3	CRS3, CRS2, CRS1, CRS4
4	CRS3, CRS2, CRS1
5	CRS3, CRS2, CRS4
6	CRS4
7	CRS3, CRS2
8	CRS3
9	CRS2
10	CRS3, CRS4

```
1.  option solver cplex;

2.  set T;
3.  set C;
4.  set F dimen 4;
5.  var x {C,T} binary;
6.  var u{T} binary;

7.   data;
8.   set T:= 1 2 3 4 5 6;
9.   set C:= 0 CRS1 CRS2 CRS3 CRS4 ;
10.  set F: 0 CRS1 CRS2 CRS3 CRS4 :=
     (CRS2,CRS1,0,0)(CRS1,0,0,0)(CRS3,CRS2,CRS1,CRS4)(CRS3,CRS2,CRS1,0)
     (CRS3,CRS2,CRS4,0)(CRS4,0,0,0)(CRS3,CRS2,0,0)(CRS3,0,0,0)(CRS2,0,0,0)(CRS3,CRS4,0,0);

11.  subject to diff{(A,B,C,D) in F, c in T}: x[A,c] + x[B,c] + x[C,c] + x[D,c] <= 1;
12.  subject to assign {c in C: c != 0}: sum {t in T} x[c,t] = 1;
13.  subject to maximum {t in T}: sum {c in C} x[c,t] <= 50;
14.  subject to condition {t in T, c in C}: x[c,t] - u[t] <= 0;

15.  minimize numOfTimeSlots: sum {t in T} u[t];

16. solve;
```

Figure 1: The representation of the variables and equations used in the formulation of phase 1 in AMPL syntax

```
1.  option solver cplex;
2.  option cplex_options 'solutionlim=2 timelimit=120';

3.  set D;
4.  set T;
5.  set J dimen 4;
6.  var y {T,D} binary;
7.  var has_2{J} binary;

8.  data;
9.  set D:= d1 d2 ;
10. set T:= 0 1 2 3 4 ;
11. set J: 0 1 2 3 4 :=
    (2,3,0,0)(3,0,0,0)(1,2,3,4)(1,2,3,0)(1,2,4,0)(4,0,0,0)(1,2,0,0)(1,0,0,0)(2,0,0,0)(1,4,0,0);

12. subject to diff{(A,B,C,D) in J, c in D}: y[A,c] + y[B,c] + y[C,c] + y[D,c]  - has_2[A,B,C,D] <= 1;
13. minimize obj: sum {(A,B,C,D) in J} has_2[A,B,C,D];
14. subject to assign {t in T}: sum {d in D} y[t,d] = 1;
15. subject to max_limit {d in D}: sum {t in T} y[t,d] <= 3;

16. solve;
```

Figure 2: The representation of the variables and equations used in the formulation of phase 2 in AMPL syntax

Table 3: The GJU data for the first 2016/2017 semester

Enrolled students	3821
Number of rooms	323
Number of exams	403

Validation based on GJU Data

The GJU registration data (shown in Table 3) for the first 2016/2017 academic semester was first used in several experiments to illustrate that the proposed method is practical i.e., fast, accurate, stable, and requires basic hardware resources.

The first experiment was conducted to find a default solution limit to use in the MyGJU Admin tool in order to reduce the time registrars need to find a feasible and acceptable final exam timetable for any academic semester. A feasible exam timetable must meet all the desired hard constraints. Whereas, a timetable is considered better (i.e., acceptable) than another one, if it results in less students having two exams in the same day and has a fewer number of days (i.e., a shorter duration).

Only 296 out of 403 exams were scheduled in this experiment due to excluding the practical exams (i.e., laboratory exams) from the offered exams list, as the practical exams at GJU usually take place in their corresponding laboratories one week ahead of the final exams period. Moreover, only 100 rooms were used after excluding laboratories and the rooms in the distant buildings from the available rooms list. The capacity of the smallest and largest used rooms were 22 and 100 respectively.

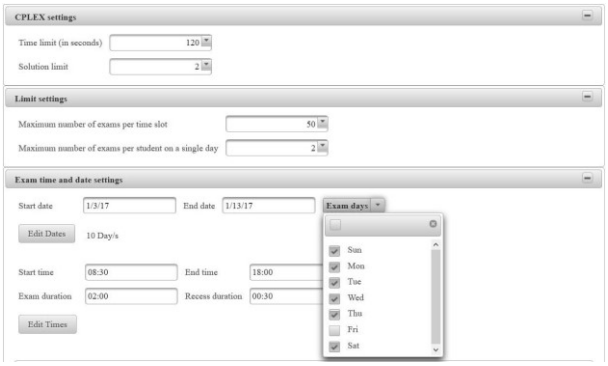

Figure 3: User interface in MyGJU Admin to adjust solver settings, exam limits, as well as exam dates and times

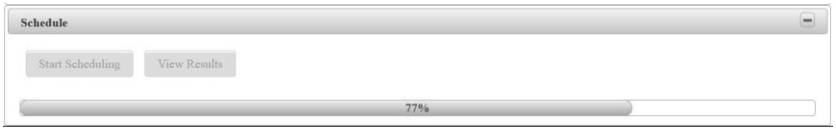

Figure 4: User interface in MyGJU Admin to launch the scheduler, monitor its progress, and then view the results

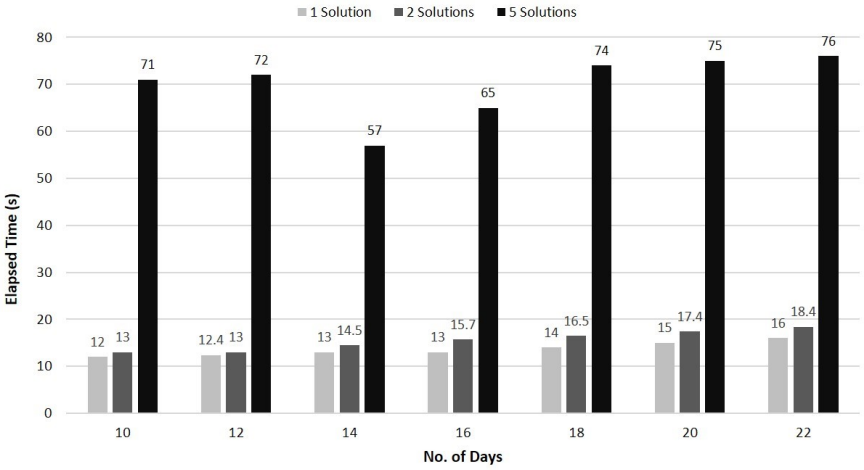

Figure 5: Elapsed times to solve the ETP in the first experiment

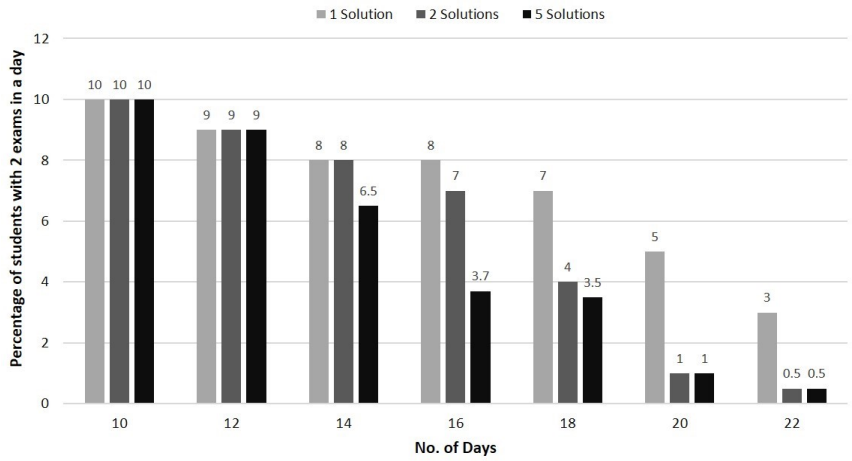

Figure 6: Percentage of students with two exams in one day in the first experiment

Table 4: Elapsed time needed to find the optimal solution, and optimal solution, for each number of days in experiment one

	10 days	12 days	14 days	16 days	18 days	20 days	22 days
Elapsed Time (s)	61420	39180	15660	5625	182	39	29
Percentage of students with 2 exams in a day	5.89	4.12	2.16	1.32	0.69	0.33	0.08

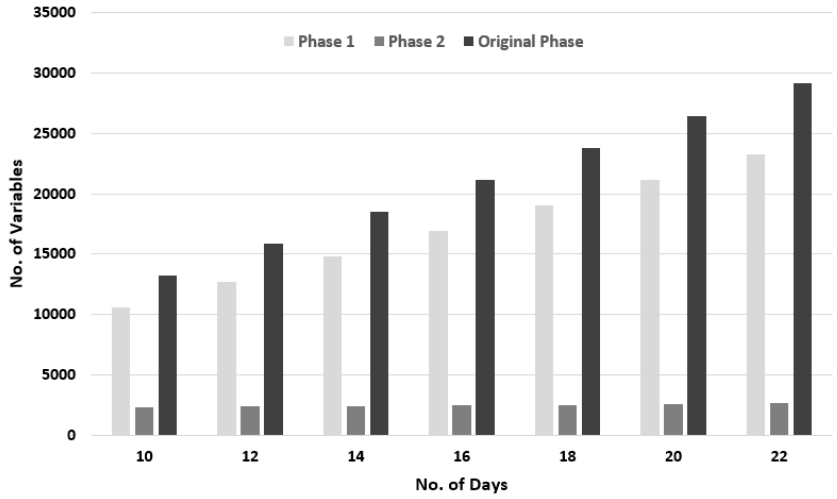

Figure 7: Number of variables in the formulations of phase 1, phase 2, and original phase

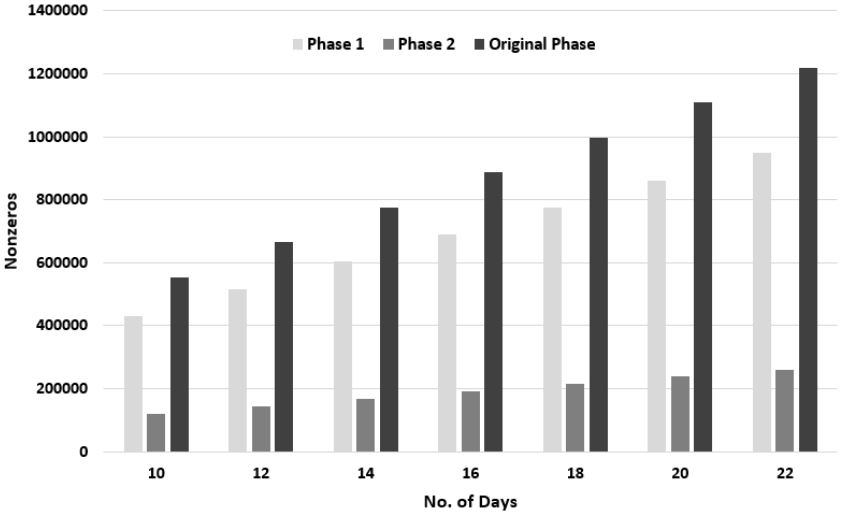

Figure 8: Number of non-zeros in the formulations of phase 1, phase 2, and original phase

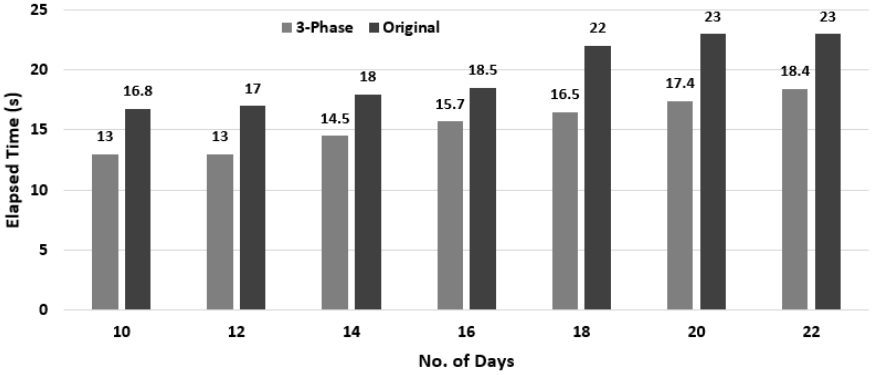

Figure 9: Elapsed times to solve the ETP using the 3-phase and 2-phase (original model) methods

Furthermore, the solver solution limit was varied as follows: 1, 2, 5, and no limit (i.e., to find the optimal solution). Also, a maximum number of different exams per time slot was set to 50. In addition, the exam days were changed as follows: 10, 12, 14, 16, 18, 20, and 22. Besides that, a

maximum of four 2 hour time slots per day (at 8:30, 11:00, 13:30, and 16:00) were allowed with a 30 minutes recess duration in between exams. Note that registrars adjusted the aforementioned solver settings (e.g., number of solutions), exam limits (e.g., maximum number of exams per student on a single day), as well as exam times and dates (e.g., start and end dates) from the final exams setup screen (see Figure 3) in the MyGJU Admin tool. While, the MyGJU Admin also supports launching the scheduler, monitoring its progress, and viewing the generated final exams timetable (see Figure 4).

The elapsed times and percentages of students with two exams in a day for the limited solutions (i.e., 1, 2, and 5 solutions) in experiment one are shown in Figure 5 and Figure 6, respectively. Whereas, the elapsed times to find the optimal (i.e., minimum) percentages of students with two exams in a day for the number of days in experiment one are given in Table 4. Accordingly, an acceptable exam timetable can be efficiently found using two solver solutions. On the other hand, finding the optimal solution might take many extra hours to slightly improve the result that was obtained with two solutions in several seconds. For example, it took about 17 hours and 6 minutes to reduce the percentage of students with two exams in a day from 10 to 5.89 for the 10 days case, which is impractical given that obtaining an acceptable percentage of 10 was achieved in about 12 seconds. This fact justifies the methodology to quickly find a feasible timetable rather than trying to find the optimal solution that mostly will take a very long time to obtain and may not be much better than the timetable that can be quickly found using two solutions.

In the second experiment, the ETP was solved using the original (2-phase) model using the setup of experiment one except for fixing the solution limit to two in all cases. Based on that, the size of the phase 1 formulation in the original model (original phase) was compared to its new counterparts (i.e., the phase 1 and phase 2 formulations in the proposed 3-phase method) based on the number of variables and non-zeros in the formulations as shown in Figure 7 and Figure 8 respectively. Whereas, the elapsed times shown in Figure 9 illustrate that the 3-phase approach outperformed the original method in all cases. Hence, the results of this experiment assert the fact that sub-dividing the ILP problem into three smaller sub-problems proved to be a key factor to enable a solver, such as CPLEX, to produce feasible solutions for an NP-complete problem in relatively short times and with reduced memory demands.

In the third experiment, the solver was used to schedule all the offered exams including the practical exams (i.e., 403 exams) using all rooms (i.e., 323 rooms). In this case, a feasible final exam timetable was found in about 42 seconds (based on one solution limit), with 9.9% of the students having two exams in the same day, and a minimum exam period of 12 days (note that an exam week at GJU usually spans six days, which brings the exams period in this case to two weeks). Hence, the practicality of the approach was proven again performance and feasibility wise although it was applied to an instance that had 107 exams more than the current number of scheduled exams at GJU.

In the fourth experiment, the dates of the GERL101 and NE101 exams were fixed as shown in Figure 10. The setup of experiment one was also used in this experiment except for fixing the solution limit to two in all cases. The result in Figure 11 illustrates that the method found a solution without altering the fixed two exam dates (i.e., it satisfied hard constraint **x**). Moreover, the results in Figure 12 and Figure 13 show that satisfying hard constraint **x** had a minor effect on the performance and accuracy of the obtained solutions.

Note that in all experiments the reported times did not only account for the time needed to find a feasible solution, but they also included preprocessing time (e.g., parsing the data files, querying the needed information from the database) as well as display time (e.g., generating output files for display). Nevertheless, the reported solution times were remarkably short and the solver did not encounter any out of memory problems while finding the solutions.

Validation based on Benchmark Data

The proposed method was further verified by using it to solve eleven real-world exam timetabling problem instances (shown in Table 5) from the University of Toronto (UT) benchmark data sets [34, 40]. The number of exams and students for each problem instance are shown in the second and third rows in Table 5, respectively. Since the UT benchmark data does not contain room information, the solver utilized the 100 GJU rooms (that were used in experiment one in the previous subsection) to assign exams into rooms in phase three. The overall capacity of those rooms is 3736 and that was enough to fit all

exams in rooms for each instance. It is also worth noting that ahead of each solution, a script was used to parse and preprocess the UT benchmark data files (i.e., the exams and student enrollments files) in order to map and import the generated output files into the corresponding database tables that will be later accessed by the ETP solver.

Manage Exams

(1 of 30) 1 2 3 4 5 6 7 8 9 10 10

Course Id	Name	Student Count	Exam days	
GERL101	German I	597	04-01-2017, Wed	
GERL201	German III	448		
NE101	National Education	412	04-01-2017, Wed	
PHYS103	Physics I	378		
GERL301	German V	365		
ENGL201	English V	359		
MILS100	Military Science	353		
MATH101	Calculus I	299		
ENGL102	English IV	279		
CS116	Computing Fundamentals	270		

Figure 10: Fixing the dates for the GERL101 and NE101 exams in the MyGJU Admin

Course Code	Course Name	Time	Date	Rooms
NE101	National Education	08:30	4-01-2017 , Wed	[B219, C208, C108, C107, H101, H401, C203, H103, H202, C202, H403, H205, H303, H204, B122, H206, B020, H505, B120, B021]
GERL101	German I	08:30	4-01-2017 , Wed	[C208, C108, H201, H302, H104, H203, H301, H103, C104, H202, C202, H304, H502, H403, H205, H303, H105, H204, H405, H206, H505, B407, B205, B222, B201, C310, B122, C210, B221, B020]

Figure 11: The fixed exam dates are not altered in the found solutions in experiment four

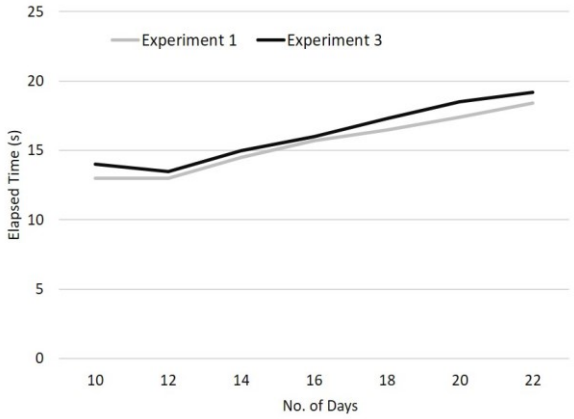

Figure 12: Elapsed times of experiments one and four with solution limit equal 2

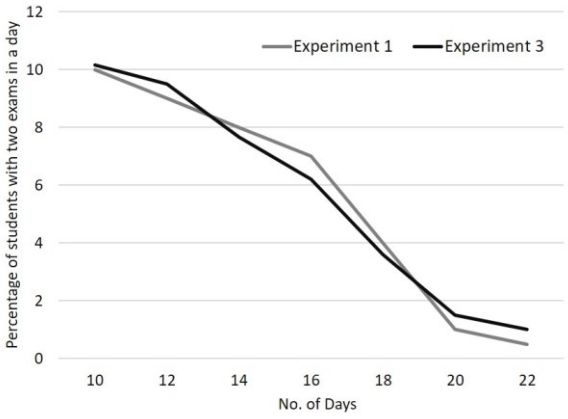

Figure 13: Percentage of students with two exams in one day in experiments one and four with solution limit equal 2

Table 5: Number of exams and students in the UT benchmark data

	Carleton91	Carleton92	EarlHaig83	EdHEC92	St.Andrews83	Trent92	TorontoAS92	TorontoE92	YorkMills83	LSE91	KingFahd93
No. of exams	682	543	190	81	139	261	622	184	182	382	462
No. of students	12990	12908	1045	1472	298	3641	13558	819	910	1772	3270

Table 6: Minimum number of days and percentages of students with 2 exams in one day for all UT instances solutions

	Carleton91	Carleton92	EarlHaig83	EdHEC92	St.Andrews83	Trent92	TorontoAS92	TorontoE92	YorkMills83	LSE91	KingFahd93
Schedule Days	13	13	12	9	8	11	13	8	12	9	12
% of students with 2 exams in one day	7.62	5.62	11.55	7.55	3.56	9.26	7.45	2.20	9.51	5.32	7.3

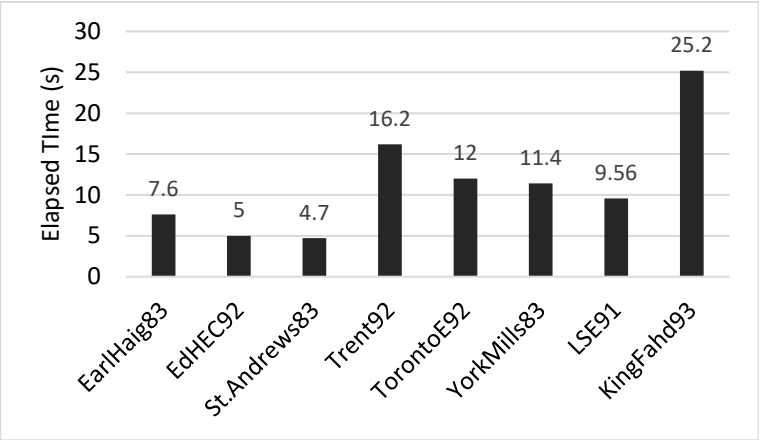

Figure 14: Elapsed times for solving the UT smaller problem instances

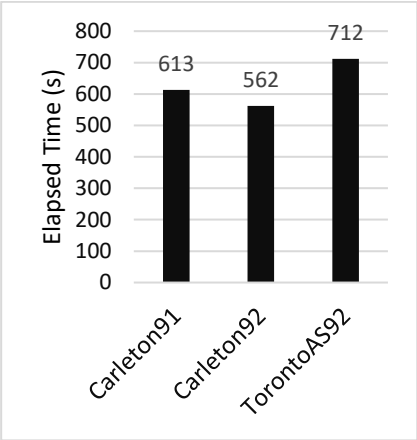

Figure 15: Elapsed times for solving the UT larger problem instances

The setup used in experiment one in the previous subsection was also used to solve each problem instance in this experiment, except for fixing the solution limit to 2 and solving for different number of days in order to find solutions that satisfy hard constraint **v** (i.e., a student cannot have more than two exams on any given day) with the least number of days. Accordingly, the minimum number of days to solve each instance while satisfying constraint **v,** as well as the percentage of students having two exams in one day for each respective solution are shown in rows two and three in Table 6, respectively. Furthermore, for readability purposes, the elapsed times to solve the corresponding smaller and larger problem instances are shown in Figure 14 and Figure 15, respectively.

Based on those results, the proposed method was capable of finding a solution for each problem instance while satisfying hard constraint **v**. Given that, satisfying constraint **v** was not considered in the first attempts to solve those cases by other researchers. Furthermore, it succeeded in finding the desired solutions despite the fact that it was performing an extra phase to assign exams to rooms. Moreover, it scheduled all exams in less than or almost two weeks. Whereas, the solutions elapsed times varied from 4.7s up to 712s (i.e., about 12 minutes) to solve the smallest (i.e., St.Andrews83 that included 298 students and 139 exams) and largest (i.e., TorontoAS92 that contained 13558 students and 622 exams) instances, respectively. Hence, it solved instances that are smaller or almost equal in size compared to the GJU instance size in a matter of seconds. While, it found solutions for instances that are almost four times the size of the GJU instance in a matter of minutes. Accordingly, the method can support the exam timetabling demands of the GJU staff for years to come with accuracy, speed, and basic hardware resources. Moreover, it can provide the majority of students with very comfortable exam schedules. As based on the percentages in Table 6, more than 90% of the students in 10 instances had a maximum of only one exam in any given day. Nevertheless, the rest of the students were also guaranteed not to have more than two exams in one day.

CONCLUSIONS & FUTURE WORK

The proposed method for solving the ETP and integrating it in the MyGJU Admin tool helped registrars at GJU maintain a comfortable examination environment, as well as meet faculty and student demands, as it enabled them to:

- Obtain a solution for the ETP using machines with basic storage and processing power.
- Control the solver and its parameters from the MyGJU Admin tool in order to easily

generate feasible final exams timetables that can be saved in various standard formats.

- Generate a feasible final exam timetable in a matter of seconds compared to several days and hundreds of iterations when they used to do it manually.
- Announce the final exam timetable several weeks ahead of the examination period, which allows instructors and students to plan ahead and prepare themselves for the exams.
- Hold some exams on specific dates, or in certain rooms, in order to meet the faculties' demands.
- Produce a comfortable timetable for all students as the method allows enforcing a hard constraint to prevent a student from having more than two exams in the same day.

The ability of the proposed method to practically solve the ETP is mainly due to decomposing it into three smaller sub-problems (phases), which drastically decreased the number of variables used in the formulation and hence reduced the demand on the processing and storage resources. Another advantage of modularizing the problem is the simplification of its formulation, which makes it readable and easier to understand.

In the near future, the skills acquired while working on the ETP will also be used to automate the course scheduling problem at GJU.

REFERENCES

1. Broder S. Final examination scheduling. Communications of the ACM 1964, 7(8): 494-498.
2. Woumans G, De Boeck L, Beliën J and Creemers S. A column generation approach for solving the examination-timetabling problem. European Journal of Operational Research 2016, 253(1): 178-194.
3. Cooper T and Kingston J. The complexity of timetable construction problems. Practice and Theory of Automated Timetabling (Lecture Notes in Computer Science) 1996, 1153: 283-295.
4. Qu R, Burke E, McCollum B, Merlot L, and Lee S. A survey of search methodologies and automated system development for examination timetabling. Journal of scheduling 2009, 12(1): 55-89.
5. Burke E, Elliman D and Weare R. A university timetabling system based on graph colouring and constraint manipulation. Journal of Research on Computing In Education 1994, 27(1): 1-18.
6. Malkawi M, Hassan M and Hassan O. A new exam scheduling algorithm using graph colouring. The International Arab Journal of Information Technology 2008, 5(1): 80-86.
7. Burke E, Kendall G and Soubeiga E. A tabu-search hyper-heuristic for timetabling and rostering. Journal of Heuristics 2003, 9: 451-470.
8. Thompson J and Dowsland K. A robust simulated annealing based examination timetabling system. Computers & Operations Research 1998, 25: 637-648.
9. Eley M. Ant algorithms for the exam timetabling problem. International Conference on the Practice and Theory of Automated Timetabling 2006, 364-382.
10. Kordalewski D, Liu C and Salvesen K. Solving an exam scheduling problem using a genetic algorithm. Department of Statistics, University of Toronto, Toronto, Canada, TR-2009-1, 2009.
11. Williams H. Logic and Integer Programming. International Series in Operations Research & Management Science, Springer, Germany, 2009.
12. Al-Yakoob S, Sherali H and Al-Jazzaf M. A mixed-integer mathematical modeling approach to exam timetabling. Computational Management Science 2010, 7(1): 19-46.
13. Arbaoui T, Boufflet J and Moukrim A. Preprocessing and an improved MIP model for examination timetabling. Annals of Operations Research 2015, 229(1): 19-40.
14. Komijan A and Koupaei M. A new binary model for university examination timetabling: a case study. Journal of Industrial Engineering International 2012, 8(1):1-7.
15. Wang S, Bussieck M, Guignard M, Meeraus A and O'Brien F. Term-end exam scheduling at United States military academy/west point. Journal of Scheduling 2010, 13(4): 375-391.
16. Kahar M and Kendall G. The examination timetabling problem at Universiti Malaysia Pahang: Comparison of a constructive heuristic with an existing software solution. European Journal of Operational Research 2010, 207(2):557-565.
17. Qu R, He F and Burke E. Hybridizing integer programming models with an adaptive decomposition approach for exam timetabling problems. Proceedings of the 4th Multidisciplinary International Scheduling: Theory and Applications 2009, 435-446.
18. Qu R, Pham N, Bai R and Kendall G. Hybridising heuristics within an estimation distribution algorithm for examination timetabling. Applied Intelligence 2015, 42(4):679-693.
19. Rahman S, Bargiela A, Burke E, Özcan E, McCollum B and McMullan P. Adaptive linear combination of heuristic orderings in constructing examination timetables. European Journal of Operational Research 2014, 232(2):287-297.
20. Müller T. Real-life examination timetabling. Journal of Scheduling 2016, 19(3):257-270.
21. Soghier A and Qu R. Adaptive selection of heuristics for assigning time slots and rooms in exam timetables. Applied Intelligence 2013, 39(2):438-450.
22. Sabar N, Ayob M, Qu R and Kendall G. Graph coloring constructive hyperheuristic for examination timetabling problems. Applied Intelligence 2011, 37(1): 1–11.
23. Burke E, Eckersley A, McCollum B, Petrovic S and Qu R. Hybrid variable neighbourhood approaches to university exam timetabling. European Journal of Operational Research 2010, 206(1):46-53.
24. McCollum, B, McMullan, P, Parkes, A, Burke, E and Qu, R. A new model for automated examination timetabling. Annals of Operations Research 2012, 194(1):291-315.
25. Qu R, Burke E and McCollum B. Adaptive automated construction of hybrid heuristics for exam timetabling

and graph colouring problems. European Journal of Operational Research 2009, 198(2):392-404.
26. Alzaqebah M and Abdullah S. Hybrid bee colony optimization for examination timetabling problems. Computers & Operations Research 2015, 54:142-154.
27. Mujuni E and Mushi A. Solving the examination timetabling problem using a two-phase heuristic: the case of Sokoine University of agriculture. Journal of Information and Computing Science 2015, 10(3): 220-227.
28. Mandal A and Kahar MNM. Solving examination timetabling problem using partial exam assignment with great deluge algorithm. IEEE International Conference on Computer, Communications, and Control Technology (I4CT) 2015, 530-534.
29. Mandal A and Kahar MNM. Solving examination timetabling problem using partial exam assignment with hill climbing search. IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE) 2015, 84-89.
30. Al-Betar M, Khader A and Doush I (2014). Memetic techniques for examination timetabling. Annals of Operations Research 2014, 218(1):23-50.
31. Akbulut A and Ylilmaz G. University exam scheduling system using graph coloring algorithm and RFID technology. International Journal of Innovation, Management and Technology 2013, 4(1):66.
32. Demeester P, Bilgin B, De Causmaecker P and Berghe G. A hyperheuristic approach to examination timetabling problems: benchmarks and a new problem from practice. Journal of Scheduling 2012, 15(1):83-103.
33. Sabar N, Ayob M, Kendall G and Qu R. Roulette wheel graph colouring for solving examination timetabling problems. COCOA 2009, 463-470.
34. Carter M, Laporte G and Lee S. Examination timetabling: algorithmic strategies and applications. Journal of Operational Research Society 1996, 47(3): 373–383.
35. Dimopoulou M and Miliotis P. Implementation of a university course and examination timetabling system. European Journal of Operational Research 2001, 130(1):202-213.
36. Al-Hawari F, Alufieshat A, Alshawabkeh M, Barham H and Habahbeh M. The software engineering of a three-tier web-based student information system (MyGJU), Computer Applications in Engineering Education 2017, 25(2):242-263.
37. F. Al-Hawari. MyGJU student view and its online and preventive registration flow. International Journal of Applied Engineering Research, 2017, 12(1):119-133.
38. F. Al-Hawari. Analysis and design of an accounting information system. International Research Journal of Electronics and Computer Engineering, 2017 Jun, 3(2):16-21.
39. Fourer R, Gay D and Kernighan B. AMPL–a modeling language for mathematical programming, Duxbury Press, USA, 2002.
40. CPLEX Optimizer. IBM. http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/, accessed 26 March 2016.
41. University of Toronto Benchmark Data. The University of Nottingham, School of Computer Science Page. http://www.cs.nott.ac.uk/~pszrq/data.htm, accessed 24 June 2017.