Examination Scheduling with Days-off Constraints

by

Azin Setayesh

Submitted in partial fulfilment of the requirements
for the degree of Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
August 2012

DALHOUSIE UNIVERSITY

DEPARTMENT OF INDUSTRIAL ENGINEERING

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled “Examination Scheduling with Days-off Constraints” by Azin Setayesh in partial fulfilment of the requirements for the degree of Master of Applied Science.

	Dated: August 7, 2012
Supervisor:	_________________________________
Readers:	_________________________________
	_________________________________
	_________________________________

DALHOUSIE UNIVERSITY

DATE: August 7, 2012

AUTHOR: Azin Setayesh

TITLE: Examination Scheduling with Days-off Constraints

DEPARTMENT OR SCHOOL: Department of Industrial Engineering

DEGREE: M.A.Sc. CONVOCATION: October YEAR: 2012

Signature of Author

پیشکش به خانواده ی دلسوزم که در این راه مرا صمیمانه تشویق و پشتیبانی نمودند.

To my loving family who have wholeheartedly supported and inspired me along the way…

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

With an increase in the number of students and the number of courses offered by universities, examination scheduling based on the available facilities becomes much more complicated. At present, examinations must be appropriately scheduled not only based on the offered courses, but also with respect to the rooms capacities, available time-slots, days-off rules and other soft and hard constraints. From the Mathematical Programming point of view, timetabling and scheduling problems are somewhat related to Assignment problems, but have additional constraints that make them computationally challenging. Timetabling problems are NP-hard problems for which there is unlikely to be an economically viable method for finding the optimal solution. In our studies, we use Python interfaced with CPLEX as optimization software to find approximately optimal results for large problems in reasonable time.

LIST OF ABBREVIATIONS AND SYMBOLS USED

GA	Genetic Algorithm
ILP	Integer Linear Programming
IP	Integer Programming
LD	Largest Degree
LP	Linear Programming
LWD	Largest Weighted Degree
SA	Simulated Annealing
SD	Saturation Degree
TS	Tabu Search
c	course
C_r	capacity of room r
E_c	number of students that take course c
g	number of time-slots
k	number of students
m	number of courses
p	number of rooms
r	room
R_{sc}	variable showing whether student s takes course c or not
s	student
t	time-slot
y	number of time-slots in one day
z	the maximum permitted number of days in a row
α	pre-defined value showing the minimum number of members of a set of students to be considered as a group (models 3 and 4)
β	pre-defined value showing the minimum number of members of a set of students to be considered as a group (models 5 and 7)

ACKNOWLEDGMENTS

I would like to thank my supervisor Dr. Pemberton Cyrus for his guidance, assistance, advice and encouragement for preparation of this research. Without his follow-ups none of this achievement would have been possible. I would also like to thank Dr. Uday Venlatadri, Dr. Ronald Pelot, and Dr. Jacek Ilow, who gave me useful advice as members of the supervisory committee.

I would also like to take this opportunity to thank my family for all the inspiration and support they provided so that I can successfully accomplish my goals.

Halifax, August 2012

Azin Setayesh

CHAPTER 1 : INTRODUCTION

This thesis looks at examination timetable scheduling for the Faculty of Engineering at Dalhousie University. Dalhousie University was established in 1818. The Technical University of Nova Scotia (TUNS) (formerly the Nova Scotia Technical College) was founded in 1907. Dalhousie and TUNS merged into a single academic institution in 1997. Also, TUNS was divided into the faculties of Engineering, Computer Science and Architecture and Planning. Currently, Dalhousie University has almost 17,000 students and m cf4ore than 1000 academic faculty and staff. The Faculty of Engineering has 1,250 students and about 130 faculty and staff.

Python is an open-source programming language, which is applied in various fields. It is available for operating systems such as Windows, Linux, and Mac OS. It can be used free even for commercial purposes. Many specialized packages are provided for python at *pypi.python.org*. Python programs are fast compared to other programming languages, and there are many integrated development environments for efficient programming.

CPLEX is commercial optimization software which is applied in various fields in order to improve efficiency. It is also available for operating systems such as Windows, Linux, and Mac OS. CPLEX optimizer provides solvers for Linear Programming (LP), Mixed Integer Linear Programming (MILP), and Quadratic Programming (QP) problems. One of the most important advantages of CPLEX is the capability to quickly solve problems with large numbers of constraints and variables. CPLEX is used in this project under a research licence.

The goal of this project is scheduling the examination timetables for the Dalhousie University community, and specifically for the Faculty of Engineering. Python is used to create an interface to the CPLEX mathematical programming solver. The case study for this project is the Faculty of Engineering at Dalhousie University, which has a desire to optimize the examination schedules. In the past, researchers have used various models and programming tools to solve examination scheduling problems and find close to optimal results. We develop and analyze seven related examination scheduling models in this research.

In section 1, we give a brief introduction to define scheduling problems. In section 2, we give a survey of previous related studies. We discuss the overall structure of this thesis in section 3 and the conclusion section presents the results of this project.

1.1. Timetabling Problem

Timetable problems are assignment activities with time-based constraints. Timetabling is used for scheduling different activities, including courses, crew assignments in airline industries, and matches in sports. In general they are referred to as scheduling problems.

For this project, we can divide timetabling in educational institutions into two major areas:

I. **School Timetabling**: In this situation, all classes for a week (or more) are scheduled, subject to instructors' resource constraints and/or students' course requirements.

II. **Examination Timetabling:** This schedules the examinations to meet room availability constraints and to avoid overlapping examination times for each student.

Furthermore, there are three major groups who directly or indirectly provide the information for the process.

a. **Administration Office:** defines the specific standards, with regards to maximum and minimum number of courses, student a permitted to select for each semester and timetabling.

b. **Departments:** may have demands for specific rooms or timeslots for their examinations. For instance, they may need to assign large examinations early in timetables to facilitate efficient marking, or they may need a room with computer facilities.

c. **Students:** usually prefer to have days off between consecutive examinations, may be concerned about the examinations' scheduling order (easy before hard examinations or vice versa), or may want to avoid consecutive hard examinations.

The primary means of assessing examination timetabling are:

1. The accuracy of the academic examination timetable.

2. Determining the stress level caused by the examination sequences.

3. Determining the cost of examination timetable improvements.

In order to properly assess the examination-scheduling requirements, it is necessary for the university administration to collect all the data in a systematic manner.

1.1.1. University Timetabling: Objective and Variants

Based on a detailed review of examination timetabling problems provided in chapter 2, most of them have the following constraints:

1. The specific examination has to be assigned to exactly one time slot.

2. No student can attend two examinations at the same time.

3. A student should not have more than one examination on the same day.

4. The number of students assigned to a room should be less than the room capacity.

Some other authors had the following specific constraints:

1. Only one examination can be held per room in a given period. This means that the examinations should be assigned to specific rooms.

2. Some examinations may need more than one timeslot.

3. Some examinations are in multiple parts, and must be held in specific consecutive periods.

4. No more than two examinations should be assigned to the same room.

5. Some of the examinations should be started after a release date and finish before a due date.

6. According to their equipment needs, some of the examinations should be held in specific rooms.

7. Consider certain examinations as a group and schedule them in the same timeslot.

8. Schedule a subgroup of examinations in a specific order.

9. Scheduling examinations with large enrolment at the beginning of the examination time period to provide enough time for marking.

10. All first- and third-year examinations take place in the morning periods and the remainder in the afternoon periods.

For our models, there are some variations to this basic problem, such as:

1. The timetable must avoid all conflicts, so that no student has more than one examination at the same time.

2. Complete all examinations in at most 16 days.

3. It must be possible to accommodate all the candidates in the various examinations rooms available. But we prefer to accommodate most of the examinations in the Sexton Gym.

4. Numbers of students assigned to each room should be equal to or less than the room capacity.

5. They should not have to take two examinations on one day.

6. The examination start time differences should be at least 24 hours, so an examination cannot be held on the morning after a previous afternoon examination.

7. Students should have one day off after three consecutive examinations, in order to have some time for recovery. This is especially important if the examinations are all challenging.

Researchers have been working on different methods and algorithms to solve examination scheduling. Some of the most frequent methods used by researchers are Sequential methods, Cluster methods, Constraint-based methods, Meta-heuristics and Integer Programming. Carter et al. (1986), Burke et al. (1998), and Di Gaspero and Schaerf (2001) are some of the researchers who have developed and applied such techniques to optimise examination scheduling.

1.2. Report Overview

This report provides a complete overview of the study, including the theoretical background, methodology, details of the coding, testing new approaches, and analyzing the results in order to provide a model based on Integer Linear Programing to schedule the exam timetable problems in a reasonable exam period length and run-time. The major stages of this thesis are:

Stage 1: collect timetable information for classes

- Review all information to collect the name and number of courses offered in the specific semester and department.
- Review all the information to collect the students enrolled for each of the courses.

Stage 2: Obtain all the classrooms actually available for examination scheduling and their capacities.

- Evaluate rooms capacities and the number of students in order to avoid splitting students registered in one course between different rooms.
- Develop an initial requirements list for place and accommodations of each examination.
- Determine the department requirements for each of the examinations.

Stage 3: Determine the available time periods for all the classrooms, which are used during the examinations.

Stage 4: Develop a methodology to find the optimal solution to the examination timetabling problem.

We have to find the optimal number of class rooms, time slots and days which can hold all the courses offered by the departments, taking into account all the constraints of the examination scheduling problem.

1.3. Problem Background

At the time this project began, Dalhousie University had assigned 3 rooms: Gym, MA120, and DalPlex, with specific capacity for all the examinations. Each day, Monday to Saturday, had three timeslots a day and almost 2000 students.

1.4. Introduction to the Technical Solution

The solution algorithm is coded in Python and embeds CPLEX. It is fed by a Microsoft Excel spreadsheet. The Excel file is a set of: 1) forms used to collect information from users as an input; and 2) a form used for output, which reports the Objective Function and variables. All the computed schedules can be saved and individually recalled at any time.

1.5. Definitions

This thesis and the literature review present many references with feasible solutions. Hard constraints are those which must be satisfied in every feasible solution. Soft constraints maybe satisfied if expedient, but can be ignored. For our problem, the feasible solution will be achieved when all hard constraints are met. Again, our criteria are that the overall schedule be balanced, the number of examinations for each student in a day should be less or equal to one, after 3 examinations in a row there should be one day off, and the number of students in a room should be less or equal to the room capacity.

1.6. Organization of Thesis

The remainder of the thesis is organized as follows. In Chapter 2, we provide an examination timetabling literature review. In Chapter 3, we explain our own model and provide and present the results of our analysis of the schedules. Finally, we draw conclusions from our work, based on the analysis.

CHAPTER 2 : LITERATURE REVIEW

2.1. Background

Examination scheduling is one of the most important and time consuming activities which takes place in an academic institution. Most research on examination timetabling has taken place during the last decade. Recently, significant research attention has been focused on both theoretical and practical aspects. The main aim of this literature review is to highlight the methodology for finding feasible and high quality results.

Carter (1986) and Laporte (1996) produced some of the most important results in examination timetabling. Many examination-timetabling researchers have used the problem collection of Carter et al. (1986) to benchmark examination-scheduling methods.

Examination timetabling problems assign a set of examinations to a limited number of time periods and rooms subject to a set of constraints. Constraints are usually categorized into soft and hard ones. For a variety of timetabling problems, Tables 1and 2 can explain some of the practical soft and hard constraints.

Table 1. Primary Hard Constraints

- No student has simultaneous examinations.
- The number of students assigned to an examination room should be equal to or less than the room capacity.

Table 2. Primary Soft Constraints

- Spread examinations as much as possible in order to not have any in consecutive time-slots or days.
- The examination of students in the same course should take place on the same day, time-slot and room.
- Schedule the examinations with high enrollment as early as possible.
- Examinations may be split over similar rooms within a time-slot.
- Only assign the same length examinations to the same room.
- Course-specific ordering of examinations needs to be satisfied.

2.1.1. Previous Surveys on Educational Timetabling

Carter (1986) researched a practical approach to graph heuristics of examination timetabling in different academic institutions. Carter and Laporte (1996) updated the previous research and examined it either on real data or implemented it for real world problems. They divided the method into four models. Cluster methods, sequential methods, meta-heuristics methods and constraint based techniques. Burke et al. (1996) and Bardadym (1996) worked on timetabling problems by gathering all the required information.

Burke and Petrovic (2002) and (2004) worked on course and examination scheduling and used algorithms including hybrid evolutionary, meta-heuristics, multi-criteria, case-based reasoning techniques and adaptive approaches. Burke and Landa Silva (2004) used a memetic algorithm in order to solve examination scheduling problems. Also, they put much effort on self-adaptive memetic algorithm design. Landa Silva et al. (2004) did research on multi-objective meta-heuristic techniques for educational timetabling problems. Their paper covered multi-phased approaches and multi-criteria evolutionary methods. McCollum (2007) has discussed both course and examination timetabling in order to fill the gap between scheduling research and practice. Schaerf and Di Gaspero (2007) proposed new approach in university timetabling. Lewis (2008) worked on a meta-heuristic method, which was applied on both course and examination timetabling. He put attention on evolutionary algorithms, ant colony simulation, Tabu search, and simulated annealing.

2.1.2. IP Models

Mir hassani (2005) worked on applying Integer Programing to examination timetabling for 324 exams per semester, 4000 students and 30 time-slots. He applied 3 types of constraint on the exams scheduled for each student, including 1) each student cannot take more than one exam in each time-slot, 2) each student cannot take more than one exam on each day, 3) each student cannot take more than one exam on any two consecutive days. Based on the students' point of view, constraint type 3 is preferred to type 2, and type 2 is preferred to type 1. The aim of this problem is to minimize the number of conflicts in the mentioned constraints, where K, L, and M $(M \ll L \ll K)$ show the students' preferred conflicts types, and $\sum_{s,d,t} Q_s \; U_{sdt}$ is the number of students with conflict type 1, $\sum_{s,d} Q_s \; V_{sd}$ is the number of students with conflict type 2, and $\sum_{s,d} Q_s \; W_{sd}$ is the number of students with conflict type 3, where Q_s shows the number of students with identical course lists:

$$\boldsymbol{Min\, Z = K \sum\nolimits_{s,d,t} Q_s \;\; U_{sdt} \;\; + L \sum\nolimits_{s,d} Q_s \;\; V_{sd} \;\; + M \sum\nolimits_{s,d} Q_s \;\; W_{sd}}$$

While satisfying the following constraints:

- Each course should have only one examination. Thus, constraint $\sum_{dt} X_{cdt} = 1$ should hold for all the courses, where $X_{cdt} = 1$ if course *c* is held on day *d* and time *t,* and 0 otherwise.
- All the students registered in course *c,* designated as GS_c *,* should be scheduled for the exam corresponding to course *c* on day *d* and time *t.* So, constraint $X_{cdt} \leq Y_{sdt}$, should hold for all students, where $Y_{sdt} = 1$ if student *s* is scheduled for an exam on day *d* and time *t,* and 0 otherwise.
- Number of conflicts of type 1 for student *s* is calculated as $\sum_{c \in GC_s} X_{cdt} \leq U_{sdt} + 1$, where GC_s is the course list of student *s*.
- Number of conflicts of type 2 for student *s* is calculated as $\sum_t Y_{sdt} \leq V_{sd} + 1$.
- Number of conflicts of type 3 for student *s* is calculated as $\sum_t Y_{sdt} + \sum_t Y_{s,d+1,t} \leq W_{sd} + 1$, where Y_{sdt} and $Y_{s,d+1,t}$, represent the number of consecutive exams for student *s.*
- Conflicts of type 1 are not allowed to happen. Thus, constraint $U_{sdt} \leq 1$ should hold for all the students on any day at any time.

The proposed method begins with a given initial schedule, and tries to minimize the mentioned types of conflicts. Thereby, finding the conflict free examination timetable, a high quality result with respect to study time for the students is obtained.

Burke et al. (2012) applied Monte Carlo hyper-heuristics in order to solve examination scheduling problems. They presented a search algorithm in order to improve the quality of the hyper-heuristic method. They applied 3 types of constraints to the exams scheduled for each student including 1) each student cannot take more than one exam in each time-slot, 2) number of students taking exams in each time-slot cannot exceed the location capacity, 3) each student cannot take more than one exam in any consecutive time-slots on each day. The objective of this problem is to minimize the weighted sum of the number of conflicts in the mentioned constraints, where w_i is the weight of conflicts of type i, and $violation(k_i, T)$ is the number of conflicts of type i for the solution T, as follows.

$$Min \sum_{i, k_i \in K} w_i \, violation(k_i, T)$$

While satisfying the following constraints:

- Each student cannot take more than one exam in each time-slot. Thus, constraint $\sum_{\forall i, e_i \in E} X_{ij} r_{ti} \leq 1$ should hold for all the students registered in course i, where $X_{ij} = 1$ if course i is held in time-slot j, and 0 otherwise, and $r_{ti} = 1$ if student t is registered for course i, and 0 otherwise.
- Number of students taking exams in each time-slot cannot exceed the location capacity. So, constraint $\sum_{i, e_i \in E} X_{ij} \sum_t r_{ti} \leq C$ should hold for all the time-slots.
- Each student cannot take more than one exam on any consecutive time-slots in each day. So, constraint $\sum_{\substack{i, e_i \in E \\ j, e_j \in E \\ i \neq j}} \sum_{u \neq L, s_u \in S} r_{ti} r_{tj} X_{iu} X_{j(u+1)} o_{u(u+1)} = 0$ should hold for all the students, where $o_{uv} = 1$ if exams u and v are scheduled on the same day, and 0 otherwise.

The proposed algorithm starts with a given initial schedule, and calculates its corresponding objective function. Then, it iteratively selects candidate schedules, and calculates their corresponding objective functions up to a point where the decrease in the value of objective function moves below a pre-defined value.

Ayob et al. (2007) worked on applying Integer Programing on examination timetables for 818 examinations, 14047 students, 75857 enrollment and 15 days with 42 available time-slots. The objective is to minimize the weighted sum of the penalty cost of taking more than one consecutive exam, where $penalty(t_i , t_j)$ is the penalty function corresponding to the exams that are scheduled in time-slots i and j, and C_{ij} is the number of students taking exams i and j.

Minimize F = $\frac{1}{M}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij} \,.\, penalty(t_i t_j)$

$$penalty(t_i t_j) = f(x) = \begin{cases} 2^{(5-\Delta t)(2-\Delta d)} , & if\ |\Delta t| \leq 5\ and\ |\Delta d| \leq 2 \\ 0, & otherwise \end{cases}$$

Where t and d show the time-slot and day of exam, respectively.

While satisfying the following constraints:

- Each course should have only one examination. Thus, constraint $\sum_{s=1}^{T} \lambda_{is} = 1$ should hold for all the courses, where λ_{is} = 1 if course i is held in time-slot s, and 0 otherwise.

- Each student cannot take more than one exam in each time-slot. Thus, constraint $\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij} \,.\, x(t_i, t_j) = 0$ should hold, where $x(t_i, t_j) = 1$ if exams i and j are held in the same time-slots, and 0 otherwise.

- Number of students taking exams at each time-slot cannot exceed the location capacity. So, constraint $\sum_{f=1}^{R} e_i \beta_{if} \leq L_f$ should hold for all the time-slots, where $\beta_{if} = 1$ if the exam i is scheduled in room f, and 0 otherwise. And L_f is the room capacity.

- The same room should be assigned to students who have consecutive exams on the same day. So, if $t_i = x, t_j = x + 1, and\ d_i = d_j, c_{ij} \neq 0\ then\ r_i = r_j$ should hold for all the exams.

- Some special examinations exclusively use the room capacity. Thus, constraint $\sum_{i=1}^{N} \propto_{ir} \leq 1$ should hold for the special exam i for all the rooms, where $\propto_{ir} = 1$ if the exam i is scheduled in room r, and 0 otherwise.

- Student cannot take more than 3 consecutive exams in the same day. So, if $c_{ij} \neq 0, c_{ik} \neq 0 , t_{ij} = x$, $[t_i = x + 1\ or\ t_j = x - 1\ ]$and $i = d_j\ then\ d_k \neq d_i$ should hold for all the exams.

- Each exam cannot be divided into more than one location. So, constraint $\sum_{f=1}^{R} \boldsymbol{\beta_{if}} = \mathbf{1}$ should hold for all the exams, where $\boldsymbol{\beta_{if}} = \mathbf{1}$ if exam *i* is scheduled in room *f*, and 0 otherwise.

They applied this model for solving the examination timetable based on a Greedy- Least Saturation Degree (G-LSD) heuristic to find high quality solutions.

Wong et al. (2002) worked on the examination timetable based on the Genetic Algorithm (GA) in order to find a solution for to 173 courses, 8800 students, and 32 exam periods. Courses are classified into three types: morning, afternoon, and night courses where $c_i = 0 \text{ or } 1\text{or } 2$ shows the course type. Similarly, the exams corresponding to each course type are categorized into the same exam-group. The objective is to minimize $\boldsymbol{H} = \boldsymbol{\alpha_1 h_1} + \boldsymbol{\alpha_2 h_2} + \boldsymbol{\alpha_3 h_3}$, where $\propto_1, \propto_2$, and $\propto_3$ are the weighting factors. Also, it counts 3 consecutive exams for all the groups by $h_1 = \sum_{k=1}^{d} \sum_{l=1}^{r} \sum_{i,j,m \in G} (s_{i,3k-3} + s_{j,3k-2} + s_{m,3k-1})$, calculates the number of consecutive night and next morning exam periods by $h_2 = \sum_{k=1}^{d} \sum_{l=1}^{r} \sum_{i,j \in G} s_{i,3k-1} s_{j,3k}$, and calculates the course class and exam period conflicts by $h_3 = \sum_{i=1}^{q} \sum_{k=1}^{d} \sum_{j=0}^{2} s_{t,3k-J} |\text{j}-c_i|$. Where s_{ik} =1 if exam *i* is scheduled at time-slot *k,* and 0 otherwise.

The following constraints are also satisfied.

- Each course should be taken in only one time-slot. Thus, constraint $\sum_{k=1}^{3d} \boldsymbol{s_{i,k}} = \mathbf{1}$ should hold for all the courses, where $\boldsymbol{s_{i,k}}$ =1 if course *i* is held in time-slot *k,* and 0 otherwise.

- Among the courses of each group, only one exam can be taken in each time-slot. Thus, constraint $\sum_{t \in Gj} \boldsymbol{s_{i,k}} \leq \mathbf{1}$ should hold for all the exam-groups and time-slots.

- No more than *l* exams are allowed to be taken in each time-slot. Thus, constraint $\sum_{i=1}^{q} \boldsymbol{s_{i,k}} \leq \boldsymbol{l_k}$ should hold for all the time-slots, where $\boldsymbol{l_k}$ is the maximum number of exams.

- Each exam can be scheduled in time-slot *k* only if that time-slot is available. So, constraint $\sum_{t\in Gj} \boldsymbol{s_{i,k}} \leq \boldsymbol{a_{i,k}}$ should hold for all the exam-groups and time-slots where $\boldsymbol{a_{i,k}} = \mathbf{1}$ if time-slot *k* is available for exam *i*.
- Some exams are already scheduled at specific time-slots. So, constraint $\boldsymbol{s_{i,k}} \geq \boldsymbol{b_{i,k}}$ should hold for all the exams and time-slots where $\boldsymbol{b_{i,k}} = \mathbf{1}$ if exam *i is* already scheduled in time-slot *k*.

Burke et al. (1999) applied an Evolutionary Algorithm to examination timetabling by minimizing the penalty cost of taking two consecutive exams on the same day. The contribution of this paper is to divide the problem into smaller problems by dividing the exam period length into several periods. The objective function is defined as following:

$$\sum_{i=1}^{E-1} \sum_{j=i+1}^{E} [\sum_{p=1}^{P} t_{ip} t_{j(p+1)} c_{ij} d_{p(p+1)} + \quad t_{ip} t_{j(p-1)} c_{ij} d_{p(p-1)}] + 5000 t_{i(p+1)}$$

Where $\boldsymbol{t_{ip}} = \mathbf{1}$ if exam *i* is assigned to period *p*, and 0 otherwise. $\boldsymbol{c_{ij}}$ shows the number of students that take both exams *i* and *j*, and d_{pq} is a penalty cost that is applied if the time-slots *p* and *q* are on the same day.

The following constraints are specified.

- Each course should be taken in only one time-slot. Thus, constraint $\sum_{p=1}^{p+1} \boldsymbol{t_{ip}} = \mathbf{1}$ should hold for all the courses.
- Each student can only take one exam in a time-slot. So, constraint $\sum_{i=1}^{E-1} \sum_{j=i+1}^{E} \sum_{p=1}^{P} t_{ip} t_{jp} c_{ij} = \mathbf{0}$ should hold for all the students.
- The number of students taking exams in each time-slot cannot exceed the location capacity. So, constraint $\sum_{i=1}^{E} \boldsymbol{t_{ip}}\ \boldsymbol{s_i} \leq \boldsymbol{S}\ \textit{should}$ hold for all the time-slots, where $\boldsymbol{s_i}$ is the number of students registered for the course *i,* and *S* shows the number of seats available.

Their method was tested on four datasets collected from Carleton University, King Fahd University, Nottingham University, and Purdue University. Number of exams to schedule were 543, 461, 800, and 2419, respectively. Enrolments were 55552, 25118, 34265, and 120690, respectively. Number of periods were 36, 21, 23, and 30, respectively. Number of seats per period were 2000, 1955, 1550, and 5000, respectively. The results indicate that the number of conflicts in each time-slot and the run-time decreased in comparison with the results obtained by solving the entire problem.

2.2. Examination Timetabling Approaches

We present these methods based on the classifications given by of Carter and Laporte, roughly divided into these four categories.

2.2.1. Sequential Methods

In examination scheduling problems, the examinations are shown by vertices in a graph and hard constraints are shown by the edges between the vertices. In the graph coloring problem, any of the contiguous vertices have different colours associated to the problem of assigning time-slots to examinations. Soft constraints should be considered separately and evaluated in order to assess the solution quality. This method consists of ordering strategies which include saturation degree (SD), largest degree (LD), largest weighted degree (LWD), and random ordering. It is worth mentioning this sequential strategy does not work well over all types of examination scheduling problems. By using the LD approach, one can schedule the examinations which have the most conflicts with others first. LWD is the same as the LD, but instead the examinations that have the most number of students and involved in conflicts are scheduled first. SD scheduled the examinations with a lower number of available periods in time-slots first. The advantage of this method is achieving reasonable results within a short run time. After building initial solutions, improvement techniques were applied. Different researchers such as Welsh and Powell (1967), Carter (1986), Burke (1998) Burke and Newall (2004), Asmuni et al. (2005) , and Corr et al. (2006) worked on this approach.

2.2.2. Cluster Methods

Clustering is a fundamental approach for data mining to find the similarities within the data without any concerns about historical knowledge. The clustering approach assesses the set of objects and forms a smaller number of groups using the similarity and factual closeness between the objects. So, the students in one cluster have more similarities among each other compared to the ones spread among other clusters. In the clustering approach, a process is a collection of iterations of assignment and centering. The centering process is the same as entropical scheduling that lies in each assignment process. Also, clustering has been applied using probabilistic assignment and distribution of assignments' probabilities that are gradually changed. This method solves examination scheduling problems in two steps. At first, it makes student clusters by hierarchical clustering and the next step is defined by putting the clusters into suitable timeslots. The aim of clustering is to avoid any immediate local minima. Moreover, the final point, a local minimum state, is compared to the nearest neighbourhood. Consequently, the final state may not be a global minimum. The method is insensitive to initial values. This method can assign the examinations into suitable timeslots effectively and it is capable of resolving the examination conflicts for each of the clusters. Different researchers such as Shunichi Shimoji and Sukhan Lee (1994),Tang Van To et al.(2010) and J. Thompson et al. (2006) have published on this approach and implementations of it.

2.2.3. Constraint-Based Methods

Constraint-based methods have attracted attention due to ease and flexibility of application in examination-scheduling problems. Based on this approach, examinations are modelled as a set of resources with defined capacities, a set of activities with defined time and resource requirements. Also, a set of hard and soft constraints is defined between the variables to assign a value to each of them to try to satisfy all the constraints. For each scheduling problem there exists a solution. If there is not a viable solution for the problems, a backtracking method is applied to find a feasible solution. An exponential increase in possible assignments has made this model computationally expensive. However, in order to decrease the computation time, different approaches are applied to this method. For instance a labelling strategy is considered as a new way that indicates the order of constituted variables. Also another approach is to obtain partial solutions, thereafter applying a local search strategy to find an improved solution. It is worth mentioning that constraint-based techniques provide a high quality result in some of the benchmark problems. Brailsford et al. (1999), Reis and Oliveira (1999), Merlot et al.

(2003) and Duong and Lam (2004) and Le Huede et al. (2006) are among the papers published in this area.

2.2.4. Meta-Heuristic Methods

Meta-heuristic methods consist of several general techniques such as Tabu search (TA), Simulated Annealing (SA) and Genetic Algorithm (GA) which can find the feasible solution. By applying different neighbourhood constructions, and moving operators among the search space detecting different local search methods is achievable. Based on the objective function, applying a meta-heuristic method can assess the quality of the examinations scheduled. The TS method investigates the search space by visiting a list of recent moves only once. Meanwhile, an aspiration strategy selects the best solution which has been generated so far unless the search moves to find other solutions in order to avoid being trapped in local optima. This method could be improve by changes of a single examination, and swapping groups of examinations followed by changing the sequence of single examinations. Glover and Laguna(1993), Di Gaspero and Schaerf (2001), Di Gaspero (2002), White and Xie (2001) and Paquete and Stutzle(2002) are some of the researchers who have examined the TS approach.

SA, which is derived from the annealing process, defines the temperature which is used in a cooling schedule to control the probability of accepting worse moves. Thus, the worse moves are considered with higher probability and they can increase the search space at the beginning of the process. The initial and final temperatures and cooling factor play an important role in the success of this approach. The various improvement methods are applied by many researchers. The method is separated into two stages in order to use the feasible solution from the first stage as an input for the SA process in the second stage to satisfy the constraints. Although the output of the first stage may be poor, the backtracking method will be applied to find a better initial solution. It is worth mentioning that the result of this method is precise, despite having considerable computational cost. Aarts and Korst (1989), Aarts et al. (2005), Thompson and Dowsland(1998), Bullnheimer (1998), Merlot et al.(2003), Duong and Lam(2004), Burke et al.(2004) and Burke and Newall(2004) are some of the researchers using the SA approach.

GA is one of the most important methods applied for examination scheduling problems. Hybridisations of GA with local search methods considered as a particular GA are named memetic algorithms. The genetic algorithm is the evolutionary process of manipulating a number of the solutions in the search space. Solutions are considered as chromosomes and developed by applying crossover and mutation operators in order to get better solutions among series of generations. Different researchers such as Corne et al. (1994), Ross et al. (1996 and 2003) , Terashima–Marin et al. (1999), Erben (2001), Wong et al.(2002), and Ulker et al. (2007) worked on this approach.

CHAPTER 3 : EXAMINATION SCHEDULING MODEL AND ANALYSIS

3.1. Problem Statement

The classical examination-scheduling problem aims to assign a certain number of examinations, associated with the same number of courses, to limited available resources provided by the examiner. The available resources include the space and time, such as the number of rooms and the number of time-slots that are available for the examination-scheduling purpose. In a resource-constrained environment, the pressure on the available rooms and time-slots has to be carefully analyzed. As the number of examinations increases, the scheduling problem gets more complicated if the level of resources remains unchanged. As the level of available resources decreases, the scheduling problem gets more complicated. Thus, the problem complexity level increases with either an increase in the number of examinations, or a decrease in the level of available resources. Consideration of the students' comfort, and an increase in the number of students, can increase the problem complexity level even further. In order to satisfy the students' comfort, the scheduling problem has to be restricted by adding some constraints. The examination-scheduling problem herein is modeled as a binary integer-programming model that is known to be NP-hard. The number of variables in this model increases with students, rooms, time slots and courses, and the associated time required to solve the problem increases exponentially.

In this study, we focused on the scheduling of the Dalhousie University examinations to check the performance of the presented approach. The problem is to assign 250 examinations, associated with up to 1400 full-time students, to 3 rooms over 16 days, each with 3 time-slots per day.

3.1.1. Scope and Purpose

We develop our model based on the assignment problem concept. The objective of this study is to develop a simplified model by formulating integer linear constraints. The model has to be able to reduce the number of conflicts in the timetabling process. For this purpose, first we have to collect the names and numbers of courses offered by the academic institutes and departments. Then we have to collect all the information on course enrolments in order to evaluate the number of registered students for each course, and the combinations of registered courses. Next, we collect the rooms information in order to assess the number of available rooms, rooms capacity and any initial requirements needed for the examinations. We will introduce a model to achieve an efficient examination timetable, based on the course lists, registration information, rooms capacity, length of the examination period (days), and the number of time-slots available on each day.

The contribution of this study is the implementation of integer linear programming software to solve examination scheduling in universities, subject to days-off constraints.

3.2. Dataset Analysis

In order to properly evaluate the facilities and future requirements, it is essential to collect all datasets in a systematic manner.

3.2.1. Space Dataset

The space dataset includes the details on all available rooms used during the examination time. All this information is obtained from Dalhousie University. The detailed dataset has the following information:

1. List of rooms available during the examination period, based on the university's preferences.
2. The capacity of each room.

In our problem, there are 3 rooms with different capacities available during the examination time: the Gym (220 students), MA121 (50 students) and DalPlex (400 students). In order to satisfy our constraints, the rooms capacities should be equal to or more than the number of students that take their examinations there.

3.2.2. Student Enrolment and Course Information

The following information was obtained from the engineering administration office. The data was entered into an Excel file.

I. The students registered in each course in the third to the fifth year for each department, and the first two years of the common (lower division) program. Students in the first two years can take courses in any of the first two years, but not in year's three to five, similarly, years three to five students cannot take courses in years one and two. In the last three years (the upper division), many students take courses across departments.

II. Names and course numbers of classes offered in all departments.

3.2.3. Time and Day Dataset

We collected the available time-slots for each room during the examination period because time is a considerable constraint. Each student can only take one examination on each day either from 8:30 to 11:30 or from 12:00 to 3:00 or from 3:30 to 6:30.

3.2.4. Constraint Analysis

To prepare the examination timetable, the following constraints must be considered:

I. The timetable must be provided in such a way that no student has more than one examination at the same time. In practice this is not difficult to achieve if enough examination rooms exist.

II. All examinations should be completed in at most 16 days. This is crucial because some time has to be allocated for processing and marking before the beginning of the next semester or the end of the academic year. The university administration has taken the position that the examination period cannot be extended.

III. It must be possible to accommodate all the candidate examinations in various available rooms. In theory, many rooms may be available, but in practice only the larger rooms are considered viable because of the ease of administration and avoidance of excessive invigilation. The University has one large hall, which can accommodate 220 students. The University prefers to assign this room to as many examinations as possible. In the case that taking an examination in the Gym is not feasible, other large lecture rooms are used as substitutes.

IV. When students are taking more than one examination per semester, ideally the examinations should be spread throughout the 16 days. In practice, this is almost impossible to be satisfied for all the students, but at the very least the student should not take two examinations on one day; setting the examination start time difference to be at least 24 hours, ensures that a student cannot have two examinations on the same day.

V. Students should have one day off after three consecutive examinations, to have some time for preparation and recovery.

3.3. Basic Solution Requirements

3.3.1. Common Software

Both the input and output files are provided in Excel spreadsheets. The model is coded in Python. The reason to use Python is that it is an open source programming language with the advantage of a user-friendly environment for which all the required modules for mathematical calculations are available in the Python online library. In order to solve the ILP problem, we used CPLEX optimization software, interfacing with Python.

3.3.2. Computer Power and Computing Time Requirements

The program runs on a standard computer which meets the RAM requirements. For all the samples which will be presented in chapter 3 and the Appendix, the problems are solved by running the program on a personal computer with an Intel® Xeon® E3-1230 CPU running at 3.30GHz, with 32Gb RAM and running 64-bit Windows 2008 server R2. In order to test the method, we start with a small sample which is shown in this chapter. The result will validate the constraints and solution method. The timetable will be considered feasible if and only if all the examinations are scheduled once. In order to avoid any student conflicts, we have to make sure that none of the students attends more than one examination in the same timeslot.

3.4. Integer linear Programming Background

Linear programming (LP) or linear optimization is a mathematical method to achieve optimal results based on the model requirements. Integer Linear Programming (ILP) is a specific form of mathematical optimization. The LP is a method to optimize the linear objective function with equality or inequality constrains. The LP model is formulated in the following form:

$$\textbf{Maximize } \boldsymbol{xC^T}$$
$$\textbf{Subject to } \boldsymbol{Ax \leq b}$$
$$\textbf{And } \boldsymbol{x \geq 0}$$

where x is the vector of variables, C and b are the coefficients' vectors and A is a matrix of coefficients. The LP solution is in an integer form if it has at least one optimal answer that is integer, and if the formulation of the problem is totally unimodular.

The LP is widely used in various fields of study. It can be used in science, economics, engineering, transportation, scheduling and manufacturing problems. Its role is considerable in modeling of planning, scheduling, and assignment problems.

These types of problems are divided into two subsets: unconstrained and constrained optimization problems. In unconstrained problems, the aim is to find decision variables that maximizes or minimizes the objective function while the decision variables are not restricted. On the other hand, the decision variables are restricted in the constrained problems. Most real world problems, same as our problem, are constrained optimization problems.

3.5. The Solution Procedures

According to Lawler (1976) the mathematical model for the assignment problem is as follows:

Minimize Z (Cost) = $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\ x_{ij}$,

where both i and j range from 1 to n [i= 1,2,…,n ; j=1,2,…,n].

x_{ij} = 1 if the i^{th} task is assigned to the j^{th} worker and x_{ij} = 0 otherwise.

Subject to the following constraints:

i. $\sum_{i=1}^{n} x_{ij}$ = 1; j=1,2,…,n

ii. $\sum_{j=1}^{n} x_{ij}$ = 1; i=1,2,…,n

$x_{ij} \geq 0$, for all i and j; also, x_{ij} is binary for all i and j.

The first constraint emphasizes that each assignee has to perform exactly one task, while the second one emphasizes each task has to be performed by exactly one assignee. If the x_{ij} integrality condition is removed, the model can be easily solved as an LP, and will give an integer solution because it meets the total unimodularity conditions.

3.5.1. Introduction

The basic concept for setting up the examination scheduling model is to fit course c into room *r* at period *t* for student *s*. The following model is a conceptual simplified cost-minimizing model. Each room cost is defined based on the user's priority.

Minimize $\sum_{s=1}^{k}\sum_{r=1}^{p}\sum_{t=1}^{g}\sum_{c=1}^{m} d_{rt}x_{srtc}$

where d_{rt} is the cost incurred when a course is assigned to be in room *r* in period *t*; also, x_{srtc} is 1 when course c is assigned to room in period *t* for student *s*, and is equal to zero otherwise.

The following constraints and assumptions are taken into account to solve the problem.

- Room capacity constraint;
- Maximum period for examinations;

- Each student should have no more than one examination on each day;
- Rooms are available from 8:30 to 11:30, 12:00 to 3:00, and 3:30 to 6:30;
- The interval between the start of examinations for each student should be at least 24 hours;
- Students should not have examinations on more than 3 consecutive days.

Finally, in order to deal with the objective function and all the constraints, the mathematical programing software packages, CPLEX add-on in Excel and LINDO, have been used for small experiments. Both are easy-to-use tools to perform a complicated linear optimization task. These two software packages were able to solve small samples to optimality. For the big problems, we used the Python programming language interfacing with the CPLEX callable library.

3.5.2. The Model

The following mathematical model and descriptions show the formulation which we developed for the problems. The objective function minimizes the cost of assigning courses to rooms at any period of time during the examination weeks. The cost incurred at the smallest room is assumed to be much greater than the bigger ones. This causes fewer numbers of students to do their examinations in that room. Thus in this study, we preferred most of the examinations to occur in the rooms with higher capacities.

Define:

- c = course , $1 \leq c \leq m$
- t = time-slot , $1 \leq t \leq g$
- r = room , $1 \leq r \leq p$
- s = student , $1 \leq s \leq k$
- m = number of courses
- g = number of time-slots
- p = number of room

- k = number of students
- y = number of timeslots in one day
- z = maximum permitted number of days in a row
- C_r = maximum capacity of each room r
- d_{rt} = Cost of taking exam in room r and at time t
- E_c = number of students enrolled in course c
- R_{sc} = registered students in course c , $\forall_c, \forall_s, R_{sc} = \begin{cases} 1, & if\ s\ registered\ in\ c \\ 0, & otherwise \end{cases}$

The first constraint restricts the number of students doing their examinations in a specific course to be less than the capacity of the room. With this constraint all the students have the same examination, in the same room. Table 3 shows the student registration based on the courses and the total number of registration for each of the courses.

$$\sum_{s=1}^{k} \sum_{c=1}^{m} x_{srtc} \leq C_r \qquad \forall r, \forall t \qquad \textbf{(1)}$$

Table 3. Sample Course Registrations

	Courses											
Student	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1		1		1		1				1		1
2					1	1					1	
3			1				1		1		1	
4		1		1		1				1		1
5				1				1			1	
6	1						1			1		1
7		1		1		1				1		1
8						1						
9						1						
10			1				1		1		1	
11			1				1		1		1	
E_c	1	3	3	4	1	6	4	1	3	4	5	4

In the second constraint, each student has to do at most one examination in each room and day in order to have no examination conflicts.

$$\sum_{r=1}^{p} \sum_{c=1}^{m} x_{srtc} \leq 1 \qquad \forall s, \forall t \qquad \textbf{(2)}$$

The third constraint provides enough study time for the students so that the time differences between the starts of examinations must be at least 24 hours. The general formulation is:

$\sum_{t=T}^{T+y-1} \sum_{r=1}^{p} \sum_{c=1}^{m} x_{srtc} \quad \leq \quad 1 \qquad \forall s, \forall T = 1,..,n$ **(3)**

In our problem, we have 3 time slots in a day, so if the student has an exam in the second time slot, as is shown in Figure 1 by considering *T*=2 and *y*=3 the next possible exam time-slot is *T*=5.

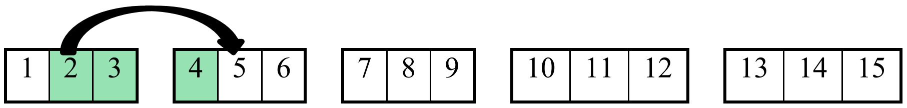


Figure 1. The Graph of Constraint 3

In our problem, we can simplify this constraint as follows:

$\sum_{t=T}^{T+2} \sum_{r=1}^{p} \sum_{c=1}^{m} x_{srtc} \quad \leq \quad 1 \qquad \forall s, \forall T = 1,..,n$ **(3a)**

Another practical condition is that for each student exactly 1 room-time will be allocated to the course he/she registered for, and 0 room-times if not.

$\sum_{r=1}^{p} \sum_{t=1}^{g} x_{srtc} \quad = \quad R_{sc} \qquad \forall c, \forall s, R_{sc} = \begin{cases} 1, & if\ s\ registered\ in\ c \\ 0, & otherwise \end{cases}$ **(4)**

The next constraint shows that the total room-time-student portion corresponding to each course is less than or equals to the total number of students registered in the course.

$\sum_{s=1}^{k} \sum_{r=1}^{p} \sum_{t=1}^{g} x_{srtc} \quad \leq \quad E_c \qquad \forall c$ **(5)**

The next practical constraint ensures that students registered in the same course do their examinations at the same time and room in order to avoid any splitting, the indices *i* and *j* represent all pairs of students.

$(x_{irtc} - x_{jrtc})\, R_{ic} R_{jc} = 0 \qquad \forall r, \forall t, \forall c, \forall (i \& j = 1, \ldots, k, i < j)$ **(6)**

The next constraint emphasizes that after three examinations are held, students have one day off. The constraint below ensures that we do not conduct more than three examinations in a row. This is performed by having no more than 3 exams in any 4-day stretch.

$\sum_{t=T}^{T+y(z+1)-1} \sum_{r=1}^{p} \sum_{c=1}^{m} x_{srtc} \leq \quad 3 \qquad \forall s, \forall T = 1, \ldots, n$ **(7)**

In our problem, we have 3 time slots in a day, so if the student has an exam in the third time slot and the student has one day off after 3 consecutive exams, as shown in figure 2 by considering T =3 and y=12 and z=3, we can have no more than 3 exams between slots 3 and 14.

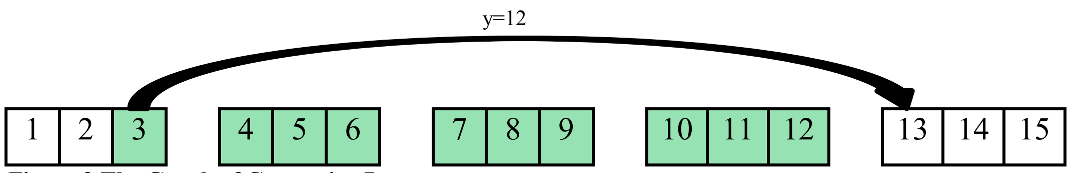


Figure 2.The Graph of Constraint 7.

Therefore, we can simplify this constraint as follows:

$$\sum_{t=T}^{T+11} \sum_{r=1}^{p} \sum_{c=1}^{m} x_{srtc} \leq 3 \qquad \forall s, \forall T=1,\ldots,n \qquad \textbf{(7a)}$$

Constraint (8) is employed (in models 6 and 7) in order to minimize the exam period length, where w represents the exam period length. We will optimize the quality of solution by taking the cost of exam period length into consideration. To do so, pw is added to the objective function, where p is the per day cost of taking exams, and is set at a significantly higher level in comparison with the rooms costs.

$$t.\sum_{r=1}^{p} \sum_{c=1}^{m} x_{srtc} \leq w \qquad \forall s, \forall t \qquad \textbf{(8)}$$

Minimize Z (Cost) = $\sum_{s=1}^{k} \sum_{r=1}^{p} \sum_{t=1}^{n} \sum_{c=1}^{m} d_r x_{srtc} + pw$

1. $\sum_{s=1}^{k} \sum_{c=1}^{m} x_{srtc} \leq C_r \qquad \forall r, \forall t$

2. $\sum_{r=1}^{p} \sum_{c=1}^{m} x_{srtc} \leq 1 \qquad \forall t, \forall s$

3. $\sum_{t=T}^{T+y-1} \sum_{r=1}^{p} \sum_{c=1}^{m} x_{srtc} \leq 1 \qquad \forall s, \forall T=\{1,\ldots,n\}$

4. $\sum_{r=1}^{p} \sum_{t=1}^{g} x_{srtc} = R_{sc} \qquad \forall c, \forall s, R_{sc} = \begin{cases} 1, & if\ s\ registered\ in\ c \\ 0, & otherwise \end{cases}$

5. $\sum_{s=1}^{k} \sum_{r=1}^{p} \sum_{t=1}^{g} x_{srtc} \leq E_c \qquad \forall c$

6. $(x_{irtc} - x_{jrtc})\ R_{ic} R_{jc} = 0 \qquad \forall r, \forall t, \forall c, \forall_{(i\ \&\ j=1,\ldots,k)}$

7. $\sum_{t=T}^{T+y(z+1)-1} \sum_{r=1}^{p} \sum_{c=1}^{m} x_{srtc} \leq 3 \qquad \forall s, \forall T=1,\ldots,n$

8. $t.\sum_{r=1}^{p} \sum_{c=1}^{m} x_{srtc} \leq w \qquad \forall s, \forall t$

3.6. Computerized Approach

We first propose the basic model for scheduling the examination timetable. Based on the output and weaknesses of our initial model, we present 6 other models. To improve execution time performance each of them has their pros and cons, which will be explained in the following sections. Models 1 to 5 use constraints 1 to 7 and ignore the examination period in the Objective function. Models 6 and 7 use the full formulation.

- Model 1 : I) ***Satisfy all the constraints for each individual student***
- Model 2 : I) ***Group students with identical course lists***

  II) Satisfy all the constraints for each group of students
- Model 3 : I) Group students with identical course lists

  II) ***Merge groups with fewer than a pre-defined number of members***

  III) Satisfy all the constraints for each group of students
- Model 4 : I) ***Divide the registration data into lower / upper divisions and for each division do the following steps:***

  i. Group students with identical course lists

  ii. Merge groups with lower than a pre-defined number of members

  iii. Satisfy all the constraints for each group of students
- Model 5 : I) Divide the registration data into lower / upper divisions and for each division do the following steps:

  i. Group students with identical course lists

  ii. Put the member of a group with lower than a pre-defined number of members into a group whose course list covers the member's course list

  iii. Satisfy all the constraints for each group of students
- Model 6: I) ***Add the cost of exam period length*** **(*pw*)** ***to the objective function, and***

its corresponding constraint* (#8) *to the constraints list

II) Follow the steps mentioned in the model 2

- Model 7: I) ***Add the cost of exam period length* (*pw*) *to the objective function, and its corresponding constraint* (#8) *to the constraints list***

  II) Follow the steps mentioned in the model 5

3.6.1. Model One – Initial Model

Error! Not a valid bookmark self-reference. shows the first model procedure, which includes all data and constraints. A model starts with getting all the registration information, which is available in the input excel file. Based on the number of days required to schedule all the examinations, which is a user-defined value, it constructs the ILP model. Then it tries to find a feasible solution by interfacing python with CPLEX. If the feasible solution is found, the model stops. Otherwise, it increases the number of days by one up to a point that a feasible solution is found.

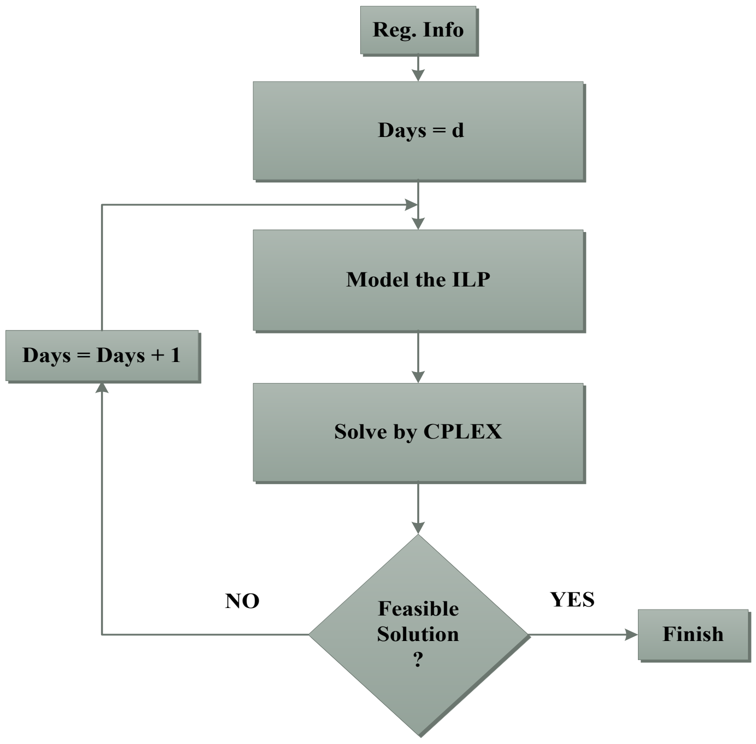


Figure 3- Initial Model

3.6.1.1. Model One Input Requirements

- **Registration Data:** Includes the registration information of all the students and their corresponding courses. It takes 1 if the student is registered the course and 0 otherwise.
- **Capacity:** shows both the capacity and adjacent of each available room.
- **Cost:** Table 4 shows the cost associated with each room at different timeslots in a day. These costs are defined based on the user preferences. According to our preference, we associate less cost to the Gym, with 220 capacity, in order to assign fewer invigilators. The Gym is relatively preferred to other rooms due to the fact that its cost per student is less in the case that some of the capacity is already occupied by other examinations. Contrary to the Gym, we assign more cost to the DalPlex since holding examinations in the DalPlex is not convenient for engineering students.

Table 4. Rooms Costs

Room\Time-slot	1	2	3
Gym	1	1	1
MA121	10	10	10
DalPlex	100	100	100

- **Course:** Includes the course names.
- **Room:** Includes the names of the rooms
- **Time:** Includes time-slots in a day.
- **Day:** Includes the minimum number of days as an initial number of days that is required for scheduling the examinations.

3.6.1.2. Model One Sample

The following ILP model is defined as a small sample, which will be used for all the models that will be described later. A feasible problem is defined by describing input, decision variables, and set of constraints that should be satisfied. In order to solve the problem in a reasonable time, we defined the sample problem considering the sample size and time complexity. The sample size determines the number of variables used in a problem. The time complexity of any algorithm shows the maximum amount of time needed for solving instances of that size. The problem size in this sample is as follows.

- Courses: 20
- Time-slots: 3
- Rooms: 3
- Students: 25

Table 5 shows the student-course registration. In this example, the maximum number of courses each student is allowed to take in each semester is six courses. Considering constraints three and seven, at least a 7 day period is required for scheduling 6 exams. As a result, the number of days is initialized at 7 days.

Table 5. Student-Course Registrations

	Course																			
Student	**ANAT1010**	**BIOL1010**	**CHEM1011**	**CLAS1100X**	**COMM1010**	**CSCI1100**	**ECED2000**	**ECON1101**	**ENGI1101**	**ENGI2200**	**ERTH1060**	**HIST1501**	**HIST1701**	**MATH1000**	**MATH1010**	**MUSC1020**	**PHYC1100X**	**POLI1020**	**STAT1060**	**THEA1000X**
1	1				1		1	1		1		1								
2		1	1	1		1			1		1									
3		1		1		1					1									
4													1		1	1	1		1	1
5		1	1			1			1		1									
6													1		1	1	1		1	1
7					1		1	1		1		1								
8			1	1					1		1									
9		1	1	1		1			1		1									
10														1	1		1			1
11	1				1		1	1		1		1								
12	1				1			1		1		1								
13															1		1			1
14													1			1	1	1	1	
15	1				1		1	1		1		1								
16	1				1		1					1								
17													1		1	1	1		1	1
18														1	1		1			1
19													1			1	1	1	1	
20		1		1		1					1									
21													1			1			1	
22	1				1		1	1		1		1								
23													1		1	1	1		1	1
24		1	1	1		1			1		1									
25	1				1		1					1								

Table 6 shows the optimal examination schedule for the above-mentioned sample model. According to Table 6, all the examinations are assigned to the Gym which is in agreement with the user cost preferences. Since the two other rooms are not assigned to any examination, their examination schedules are not shown here. The runtime is 9.5 hours. Using the Initial model, all the courses are scheduled in a 7-day exam period.

Table 6. Examinations Schedule using the First Model

			ANAT1010	BIOL1010	CHEM1011	CLAS1100X	COMM1010	CSCI1100	ECED2000	ECON1101	ENGI1101	ENGI2200	ERTH1060	HIST1501	HIST1701	MATH1000	MATH1010	MUSC1020	PHYC1100X	POLI1020	STAT1060	THEA1000X
Gym	**M**	**8:30-11:30**		*			*												*			
		12:00 -3:00																				
		3:30-6:30																				
	T	**8:30-11:30**				*			*							*					*	
		12:00 -3:00																				
		3:30-6:30																				
	W	**8:30-11:30**						*		*							*					
		12:00 -3:00																				
		3:30-6:30																				
	R	**8:30-11:30**																				
		12:00 -3:00																				
		3:30-6:30																				
	F	**8:30-11:30**										*	*									
		12:00 -3:00																				
		3:30-6:30													*							
	S	**8:30-11:30**			*									*								
		12:00 -3:00																				
		3:30-6:30																*				
	M	**8:30-11:30**																				
		12:00 -3:00																				
		3:30-6:30	*								*									*		*

3.6.2. Model Two – Grouping Model

In order to decrease the problem size, we construct the Grouping model. A group is defined as a set of students doing an identical course set. Figure 4 shows the second model procedure. As it is shown, after collecting all information, the model groups the students based on their course similarities. After grouping, all other steps are followed in the same way as the initial model. In general, a group is composed of students who enrol in the same courses. Then, each group is treated in the same way as a student in the first model. The advantage of this model is not apparent in the small samples but in the case of problems dealing with real datasets, it considerably reduces the number of variables in comparison with the direct ILP formulation.

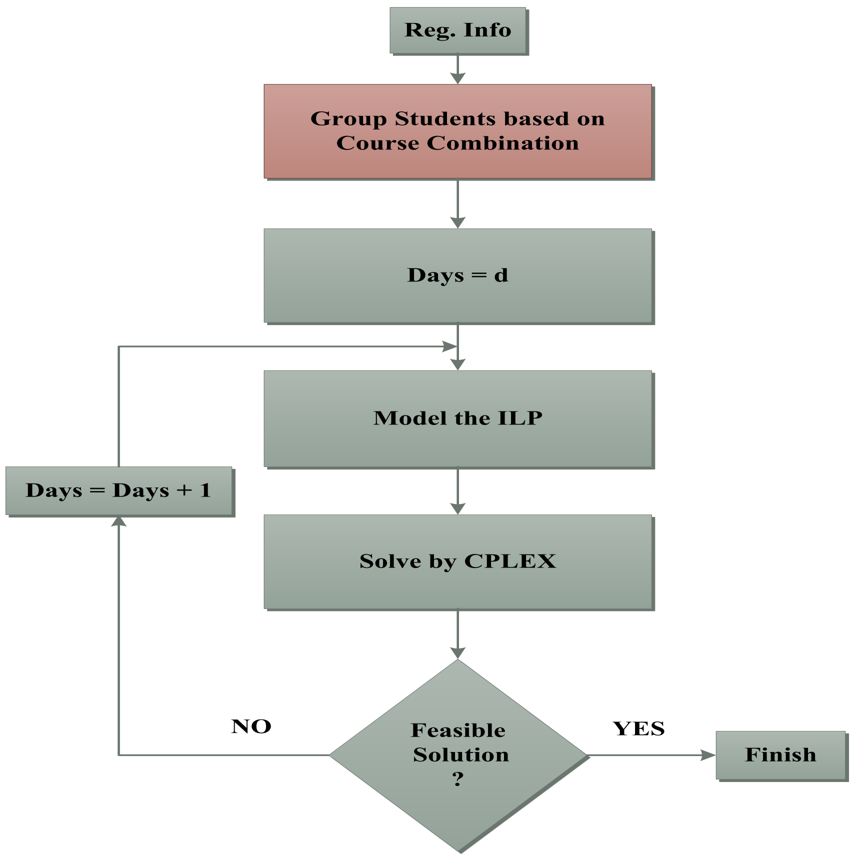


Figure 4- Grouping Model

3.6.2.1. Model Two Input Requirements

For this model, as with the Initial model, we need the Registration, Capacity, Cost, Course, Room, Time, and day information in order to provide the applicable input.

3.6.2.2. Model Two Sample

Table 7 shows the student-course registration which will be later categorized based on the course combinations. Rows shown in the same color correspond to the students whose course lists are identical. For example, students 0, 10, 14, and 21, who register in the courses ANAT 1010, COMM 1010, ECED 2000, ECON 1101, ENGI 2200, and HIST 1501, are categorized into one group. As another example, student 4, whose list of registered courses is not identical to any other students, individually forms a group.

Table 7. Student-Course Registrations (before applying the grouping approach)

Student \ Course	ANAT1010	BIOL1010	CHEM1011	CLAS1100X	COMM1010	CSCI1100	ECED2000	ECON1101	ENGI1101	ENGI2200	ERTH1060	HIST1501	HIST1701	MATH1000	MATH1010	MUSC1020	PHYC1100X	POLI1020	STAT1060	THEA1000X
0	1				1		1	1		1		1								
1		1	1	1		1			1		1									
2		1		1		1					1									
3													1		1	1	1		1	1
4		1	1			1			1		1									
5													1		1	1	1		1	1
6					1		1	1		1		1								
7			1	1					1		1									
8		1	1	1		1			1		1									
9														1	1		1			1
10	1				1		1	1		1		1								
11	1				1			1		1		1								
12															1		1			1
13													1			1	1	1	1	
14	1				1		1	1		1		1								
15	1				1		1					1								
16													1		1	1	1		1	1
17														1	1		1			1
18													1			1	1	1	1	
19		1		1		1					1									
20													1			1			1	
21	1				1		1	1		1		1								
22													1		1	1	1		1	1
23		1	1	1		1			1		1									
24	1				1		1					1								

Table 8 shows the group-course registration resulting from applying the grouping approach. It shows the list of registered courses for each group.

Table 8. Group-Course Registrations (after applying the grouping approach)

	Course																			
Group	ANAT1010	BIOL1010	CHEM1011	CLAS1100X	COMM1010	CSCI1100	ECED2000	ECON1101	ENGI1101	ENGI2200	ERTH1060	HIST1501	HIST1701	MATH1000	MATH1010	MUSC1020	PHYC1100X	POLI1020	STAT1060	THEA1000X
1	1				1		1	1		1		1								
2		1	1	1		1			1		1									
3		1		1		1					1									
4													1		1	1	1		1	1
5		1	1			1			1		1									
6					1		1	1		1		1								
7			1	1					1		1									
8														1	1		1			1
9	1				1			1		1		1								
10															1		1			1
11													1			1	1	1	1	
12	1				1		1					1								
13													1			1			1	

Table 9 shows the members of the groups. Table 10 shows the optimal examination schedule for the group-course registration shown in table 9. As is shown in Table 10, all the examinations are assigned to the Gym, which is in agreement with the user cost preferences. Since the two other rooms are not assigned to any examination, their examination schedules are not shown here.

Table 9. Members of the Groups using the Second Model

Groups	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6
Number of Members	4	3	2	4	1	1
Member	0-10-14-21	1-8-23	2-19	3-5-16-22	4	6

Groups	Group 7	Group 8	Group 9	Group 10	Group 11	Group 12	Group 13
Number of Members	1	2	1	1	2	2	1
Member	7	9-17	11	12	13-18	15-24	20

Table 10. Examinations Schedule using the Second Model

		ANAT1010	BIOL1010	CHEM1011	CLAS1100X	COMM1010	CSCI1100	ECED2000	ECON1101	ENGI1101	ENGI2200	ERTH1060	HIST1501	HIST1701	MATH1000	MATH1010	MUSC1020	PHYC1100X	POLI1020	STAT1060	THEA1000X
M	8:30-11:30		*			*												*			
	12:00 -3:00																				
	3:30-6:30																				
T	8:30-11:30				*			*									*				
	12:00 -3:00																				
	3:30-6:30																				
W	8:30-11:30						*		*										*		*
	12:00 -3:00																				
	3:30-6:30																				
R	8:30-11:30																				
	12:00 -3:00																				
	3:30-6:30																				
F	8:30-11:30										*	*									
	12:00 -3:00																				
	3:30-6:30															*					
S	8:30-11:30			*									*								
	12:00 -3:00																				
	3:30-6:30													*							
M	8:30-11:30																				
	12:00 -3:00																				
	3:30-6:30	*								*					*					*	

It is worth mentioning that by applying the grouping model, the number of variables decreased in comparison with the first model. The variables associated with the original 25 students are substituted with 13 variables associated with the newly defined groups shown in Table 8. Although the decrease in the number of variables seems to be insignificant for the mentioned sample problem, it will significantly affect the large-size problems with real datasets. Using the grouping model, the model is solved in 1.19 hours which is around 8 times lower than the time spent for the first model. Interestingly, based on the second model, the examinations are scheduled in 7 days, which is exactly the same as with the first model. Therefore, the second model provides the same quality results with the advantage of lower running time.

3.6.3. Model Three – Grouping by Considering Alpha

In the third model, students are grouped based on their course combinations in a way similar to the second model. The only difference is that, after constructing the groups, those whose number of members is lower than a pre-defined value, are merged together. The pre-defined value, called alpha, represents the minimum number of students required to form a group. Merging the groups with lower than alpha members is performed in order to decrease the problem size. Figure 5 shows the third model procedure when considering the alpha value. All the remaining steps are the same as the previous model.

3.6.3.1. Model Three Input Requirements

In this model, aside from the common data used in the Initial model, the alpha parameter is defined. For our sample, we consider alpha as equal to 3.

3.6.3.2. Model Three Sample

Applying the third model, for the data shown in Table 9, the groups 3, 5, 6, 7, 8, 9, 10, 11, 12, and 13, whose number of members is lower than 3, are merged together. As a result, the 13 former groups are substituted with 4 groups, from which one group consists of the members of 10 former groups with lower than 3 members. Table 11 shows the members of the groups using the third model approach. The list of registered courses for the newly defined group also consists of the courses registered by the members of the 10 former groups. Table 12 shows the list of registered courses for each group using the third model approach.

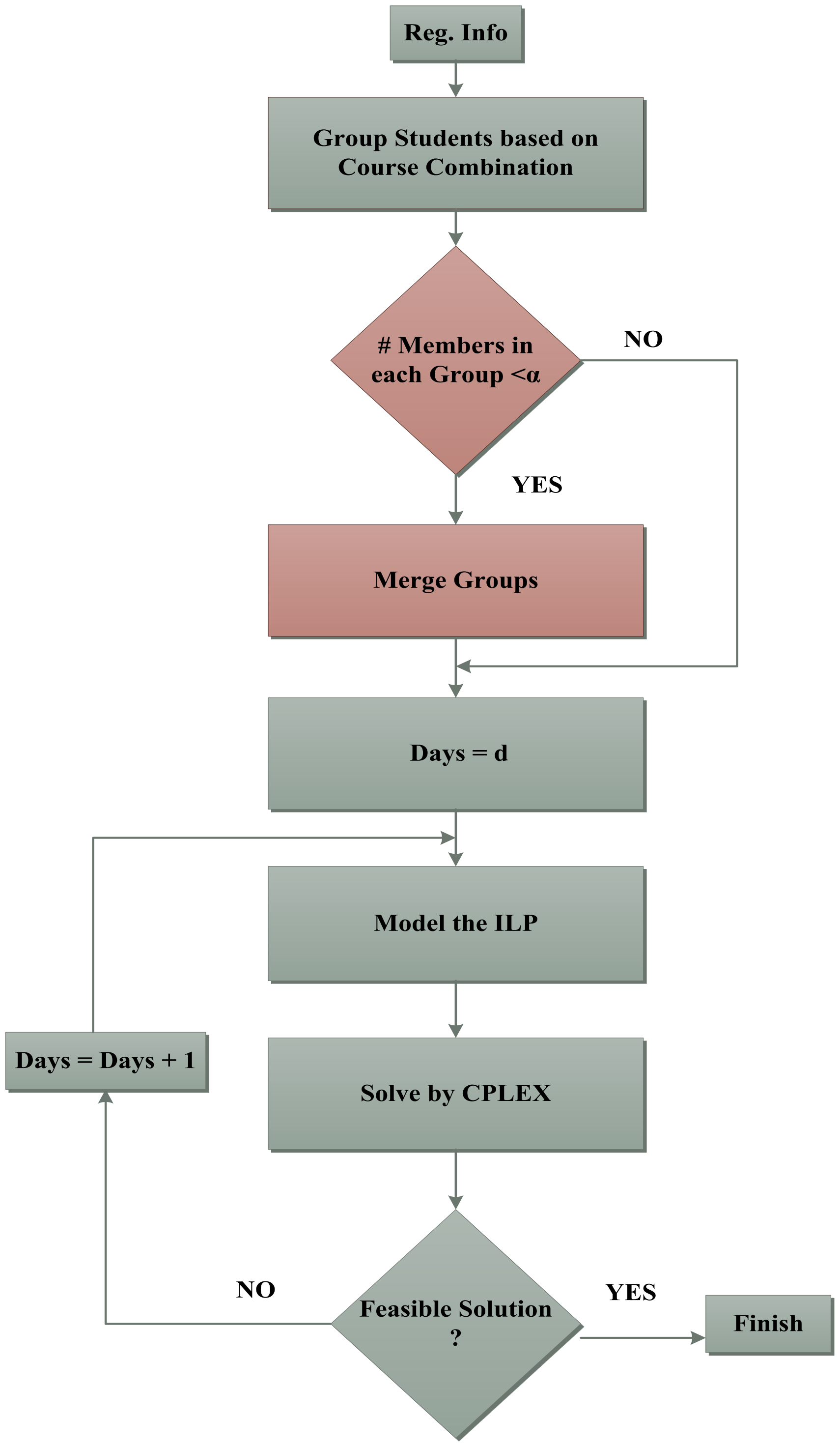


Figure 5- Grouping Model by Considering Alpha

Table 11. Members of the Groups using the Third Model (Considering Alpha=3)

Groups (First Department)	Group 1	Group 2	Group 3	Group 4
Number of Members	4	3	4	14
Member	0-10-14-21	1-8-23	3-5-16-22	2-19-4-6-7-9-17-11-12-13-18-15-24-20

Table 12. Course Selection for Each Group using the Third Model

	Course																			
Group	ANAT1010	BIOL1010	CHEM1011	CLAS1100X	COMM1010	CSCI1100	ECED2000	ECON1101	ENGI1101	ENGI2200	ERTH1060	HIST1501	HIST1701	MATH1000	MATH1010	MUSC1020	PHYC1100X	POLI1020	STAT1060	THEA1000X
1 (0-10-14-21)	1				1		1	1		1		1								
2 (1-8-23)		1	1	1		1			1		1									
3 (3-5-16-22)													1		1	1	1		1	1
4 (2-4-6-7-9-11-12-13-15-17-18-19-20-24)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 13 shows the optimal examination schedule for the group-course registration shown in the table 12. As is shown in Table 13, all the examinations are assigned to the Gym. Since the two other rooms are not assigned to any examination, their examination schedules are not shown here. It is worth mentioning that by applying the grouping model considering the alpha value, the number of variables decreased in comparison with the first and second model. The variables associated with the original 25 students were substituted with variables associated with 4 newly defined groups shown in Table 12. The decrease in the number of variables is considerable for the mentioned sample problem in comparison with the first model. However it will more significantly affect the number of variables in the large-size problems, which deal with real datasets. By using this approach, the model was solved in 5 hours, which is almost 2 times less than the time spent for the first model. Based on the first and second model, the examinations are scheduled in a 7-day period, but the third model uses a 26-day period. The reason is that the group, which is constructed as the combination of groups with fewer than alpha members, has more courses to be scheduled simultaneously. This forces the model to schedule the timetable in a longer period due to the presence of the seventh constraint, which emphasized having one day off after three consecutive examinations. Therefore, the quality of the third model decreases with respect to the scheduled examination days. But by using the third model we will be able to get the advantage of lower running times for large datasets.

Table 13. Examinations Schedule using the Third Model

			ANAT1010	BIOL1010	CHEM1011	CLAS1100X	COMM1010	CSCI1100	ECED2000	ECON1101	ENGI1101	ENGI2200	ERTH1060	HIST1501	HIST1701	MATH1000	MATH1010	MUSC1020	PHYC1100X	POLI1020	STAT1060	THEA1000X
Gym	M	8:30-11:30	*																			
		12:00 -3:00																				
		3:30-6:30																				
	T	8:30-11:30		*																		
		12:00 -3:00																				
		3:30-6:30																				
	W	8:30-11:30			*																	
		12:00 -3:00																				
		3:30-6:30																				
	R	8:30-11:30																				
		12:00 -3:00																				
		3:30-6:30																				
	F	8:30-11:30				*																
		12:00 -3:00																				
		3:30-6:30																				
	S	8:30-11:30					*															
		12:00 -3:00																				
		3:30-6:30																				
	M	8:30-11:30						*														
		12:00 -3:00																				
		3:30-6:30																				
	T	8:30-11:30																				
		12:00 -3:00																				
		3:30-6:30																				
	W	8:30-11:30							*													
		12:00 -3:00																				
		3:30-6:30																				
	R	8:30-11:30								*												
		12:00 -3:00																				
		3:30-6:30																				
	F	8:30-11:30									*											
		12:00 -3:00																				
		3:30-6:30																				
	S	8:30-11:30																				
		12:00 -3:00																				
		3:30-6:30																				
	M	8:30-11:30										*										
		12:00 -3:00																				
		3:30-6:30																				
	T	8:30-11:30											*									
		12:00 -3:00																				
		3:30-6:30																				
	W	8:30-11:30												*								
		12:00 -3:00																				
		3:30-6:30																				
	R	8:30-11:30																				
		12:00 -3:00																				
		3:30-6:30																				
	F	8:30-11:30													*							
		12:00 -3:00																				
		3:30-6:30																				
	S	8:30-11:30														*						
		12:00 -3:00																				
		3:30-6:30																				
	M	8:30-11:30															*					
		12:00 -3:00																				
		3:30-6:30																				
	T	8:30-11:30																				
		12:00 -3:00																				
		3:30-6:30																				
	W	8:30-11:30																*				
		12:00 -3:00																				
		3:30-6:30																				
	R	8:30-11:30																	*			
		12:00 -3:00																				
		3:30-6:30																				
	F	8:30-11:30																		*		
		12:00 -3:00																				
		3:30-6:30																				
	S	8:30-11:30																				
		12:00 -3:00																				
		3:30-6:30																				
	M	8:30-11:30																			*	
		12:00 -3:00																				
		3:30-6:30																				
	T	8:30-11:30																				
		12:00 -3:00																				
		3:30-6:30																				*

3.6.4. Model Four - Separating Based on the Departments

In the fourth model, students are grouped based on their course combinations exactly in the same way as the third model. The only difference is that, before the grouping phase, the model divides the students based on their corresponding departments. This way, the problem is divided into several smaller problems. Since the problems are disjoint relative to students and courses, they can be solved one by one. For each department, this model constructs the corresponding ILP model in the same way as the third model. It goes through departments one by one, and schedules each department independently up to a point where no more departments are left unscheduled. Since all the departments consume the shared room resources, the algorithm modifies the rooms capacity based on the number of students allocated to each occupied room each time a department is scheduled. Figure 6 shows the fourth model procedure with a separation approach based on upper and lower divisions. The advantage of this model is not apparent for small samples, but in the case of large input and real datasets, reduction of the number of variables is significant. Although the decrease in the number of variables resulted in a significant drop in the run time of the problem, the quality of the final solution is severely reduced. We note here that separating based on departments is an approximation, because rooms are shared between departments.

3.6.4.1. Model Four Input Requirements

- **Registration:** Consider the first and second year students in the first group, and third, fourth and fifth year students in the second group. The first and second sheets include the registration information of all the students in the first and second departments respectively and their corresponding courses.
- **Departments:** Includes the names of the departments
- **Capacity:** Shows both the capacity and adjacency of each available room.
- **Cost:** Shows the cost associated with each room at different timeslots in a day.
- **Course:** Includes the course names for each department.
- **Room:** Includes names of the rooms.

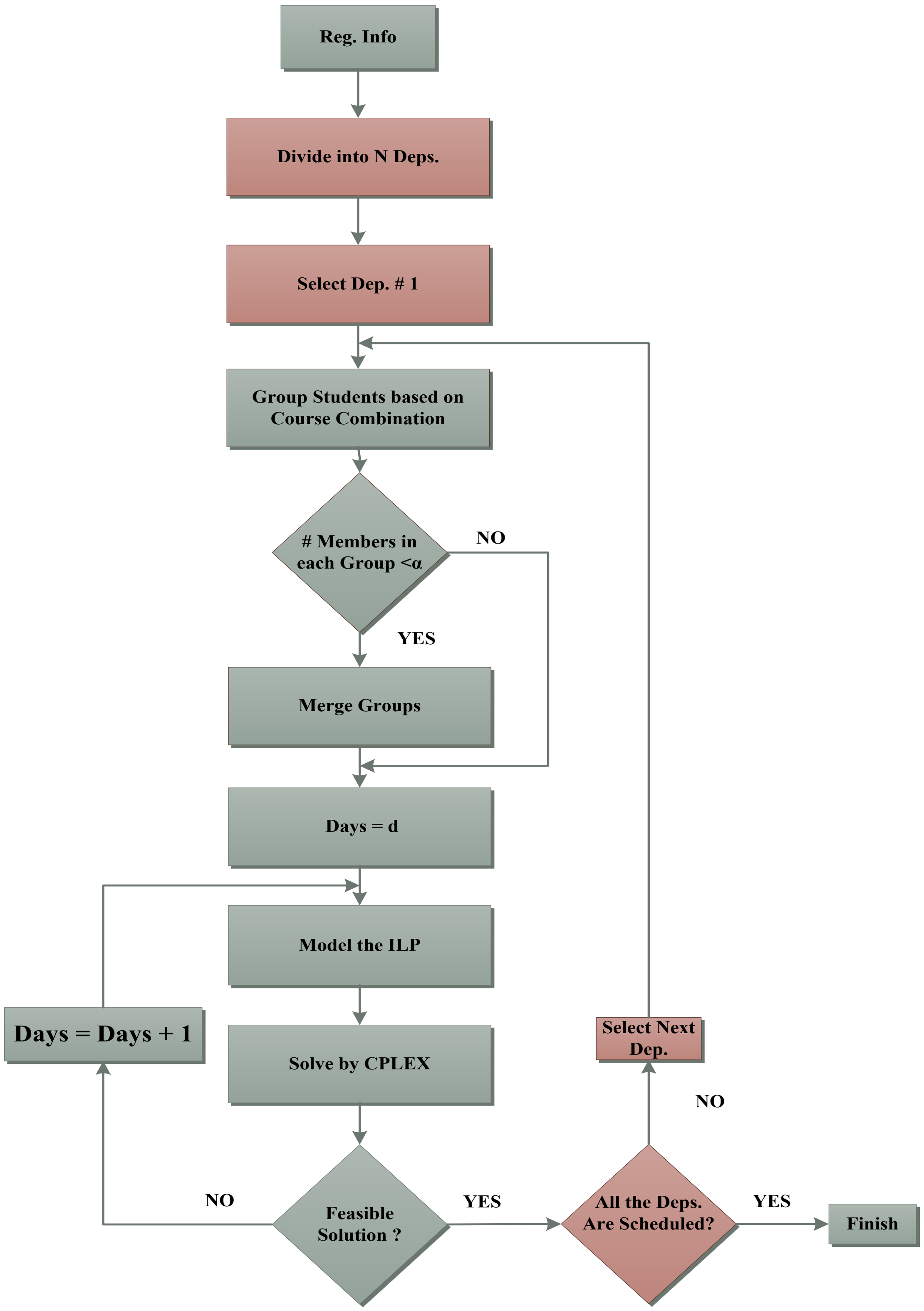


Figure 6. Separation based on Departments

- **Time:** Includes timeslots in a day.
- **Day:** Includes the minimum number of days as an initial number of days that is required for scheduling the examinations.
- **Alpha:** Includes the minimum number of students required to form a group. It is treated the same as alpha in the second model, which merges with other groups to decrease the size of the problem.

3.6.4.2. Model Four Sample

We break our previous input into two departments. The first one includes the first and second year students, and the next one includes the remaining year students. The list of students and offered courses for each department is shown in Table 14.

Table 14. The List of Students and Offered Courses in each Department

Departments	Students	Courses											
First Department	0,1,2,4,6,7,8,10,11,14,15,19,21,23,24	ANAT1010	BIOL1010	CHEM1011	CLAS1100X	COMM1010	CSCI1100	ECED2000	ECON1101	ENGI1101	ENGI2200	ERTH1060	HIST1501
Second Department	3,5,9,12,13,16,17,18,20,22	HIST1701	MATH1000	MATH1010	MUSC1020	PHYC1100X	POLI1020	STAT1060	THEA1000X				

The list of students and their corresponding registered courses for the first and second departments are shown in the Tables 15 and 16, respectively. Since the students and courses associated with different departments do not overlap each other, each department's courses are scheduled independently. Applying the grouping approach introduced in the third model, students associated with the first and second departments are categorized into 3 and 2 groups, respectively.

Table 15. Student-Course Registrations of the First Department (before applying the grouping approach)

	Course											
Student	**ANAT1010**	**BIOL1010**	**CHEM1011**	**CLAS1100X**	**COMM1010**	**CSCI1100**	**ECED2000**	**ECON1101**	**ENGI1101**	**ENGI2200**	**ERTH1060**	**HIST1501**
0	1				1		1	1		1		1
1		1	1	1		1			1		1	
2		1		1		1					1	
4		1	1			1			1		1	
6					1		1	1		1		1
7			1	1					1		1	
8		1	1	1		1			1		1	
10	1				1		1	1		1		1
11	1				1			1		1		1
14	1				1		1	1		1		1
15	1				1		1					1
19		1		1		1					1	
21	1				1		1	1		1		1
23		1	1	1		1			1		1	
24	1				1		1					1

Table 16. Student-Course Registrations of the Second Department (before applying the grouping approach)

	Course							
Student	**HIST1701**	**MATH1000**	**MATH1010**	**MUSC1020**	**PHYC1100X**	**POLI1020**	**STAT1060**	**THEA1000X**
3	1		1	1	1		1	1
5	1		1	1	1		1	1
9		1	1		1			1
12			1		1			1
13	1			1	1	1	1	
16	1		1	1	1		1	1
17		1	1		1			1
18	1			1	1	1	1	
20	1			1			1	
22	1		1	1	1		1	1

Tables 17 and 18 show the list of registered courses associated with the students in each group of the first and second departments, respectively, using the fourth model approach.

Table 17. Course Selection for Each Group of the First Department using the Fourth Model

	Course											
Group	ANAT1010	BIOL1010	CHEM1011	CLAS1100X	COMM1010	CSCI1100	ECED2000	ECON1101	ENGI1101	ENGI2200	ERTH1060	HIST1501
1(0-10-14-21)	1				1		1	1		1		1
2(1-8-23)		1	1	1		1			1		1	
3(2-4-6-7-11-15-19-24)	1	1	1	1	1	1	1	1	1	1	1	1

Table 18. Course Selection for Each Group of the Second Department using the Fourth Model

	Course							
Group	HIST1701	MATH1000	MATH1010	MUSC1020	PHYC1100X	POLI1020	STAT1060	THEA1000X
1(3-5-16-22)	1		1	1	1		1	1
2(9-12-13-17-18-20)	1	1	1	1	1	1	1	1

Tables 19 and 20 show the members of the groups shown in Tables 17 and 18, respectively. For instance, the first group in the first department has 4 members including students' number 0, 10, 14 and 21.

Table 19. Members of the Groups of the First Department using the Fourth Model (Considering Alpha=3)

Groups	Group 1	Group 2	Group 3
Number of Members	4	3	8
Members	0-10-14-21	1-8-23	2-4-6-7-11-15-19-24

Table 20. Members of the Groups of the Second Department using the Fourth Model (Considering Alpha=3)

Groups	Group 1	Group 2
Number of Members	4	6
Members	3-5-16-22	9-12-13-17-18-20

Table 21 shows the optimal examination schedule for the group-course registrations shown in Tables 19 and 20. As is shown in Table 21, all the examinations are assigned to the Gym, which is in agreement with the user cost preferences. Table 22 shows the optimal examination schedule for each group at different days and timeslots. It is worth mentioning that by applying the fourth model approach, the number of variables decreased in comparison with the first and second model, and the runtime decreased in comparison with the third model. All the variables associated with the original 25 students were substituted with 2 problems each consisting of 3 and 2 variables associated with the groups in the first and second departments, respectively, as shown in Tables 19 and 20.

Table 21. Examinations Schedule using the Fourth Model

		ANAT1010	BIOL1010	CHEM1011	CLAS1100X	COMM1010	CSCI1100	ECED2000	ECON1101	ENGI1101	ENGI2200	ERTH1060	HIST1501	HIST1701	MATH1000	MATH1010	MUSC1020	PHYC1100X	POLI1020	STAT1060	THEA1000X
M	8:30-11:30	*												*							
	12:00 -3:00																				
	3:30-6:30																				
T	8:30-11:30		*												*						
	12:00 -3:00																				
	3:30-6:30																				
W	8:30-11:30			*												*					
	12:00 -3:00																				
	3:30-6:30																				
R	8:30-11:30																				
	12:00 -3:00																				
	3:30-6:30																				
F	8:30-11:30				*												*				
	12:00 -3:00																				
	3:30-6:30																				
S	8:30-11:30					*												*			
	12:00 -3:00																				
	3:30-6:30																				
M	8:30-11:30						*												*		
	12:00 -3:00																				
	3:30-6:30																				
T	8:30-11:30																				
	12:00 -3:00																				
	3:30-6:30																				
W	8:30-11:30							*													
	12:00 -3:00																				
	3:30-6:30																			*	
R	8:30-11:30								*												
	12:00 -3:00																				
	3:30-6:30																				*
F	8:30-11:30									*											
	12:00 -3:00																				
	3:30-6:30																				
S	8:30-11:30																				
	12:00 -3:00																				
	3:30-6:30																				
M	8:30-11:30										*										
	12:00 -3:00																				
	3:30-6:30																				
T	8:30-11:30																				
	12:00 -3:00																				
	3:30-6:30											*									
W	8:30-11:30																				
	12:00 -3:00																				
	3:30-6:30												*								

Although the decrease in the number of variables seems to be considerable for the sample problem in comparison with the first model, it will more significantly affect the number of variables in the large-size problems, which deal with the real datasets.

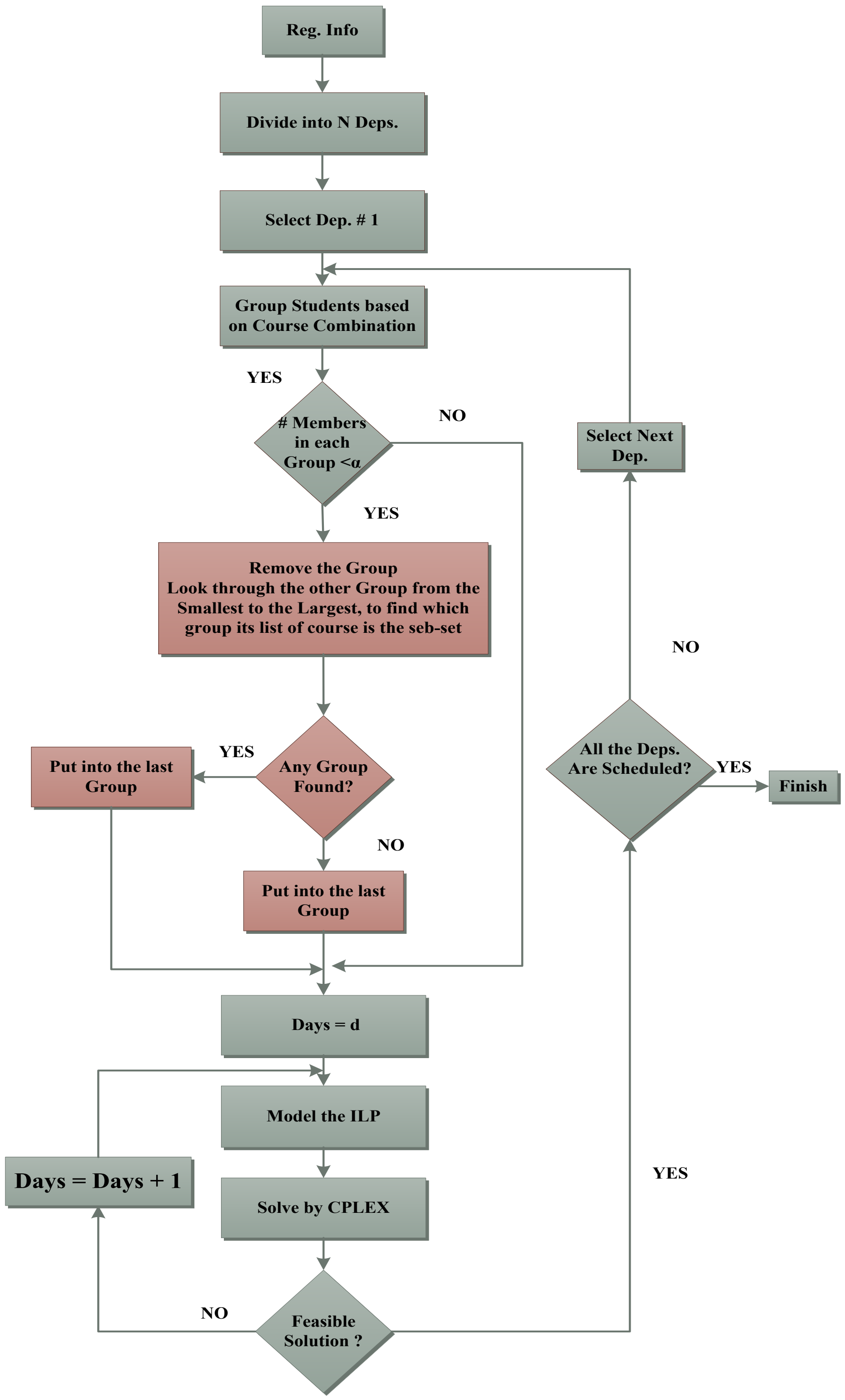
Table 22. Examinations Schedule for each Group using the Fourth Model

		Group 1-First	Group 2-First	Group 3-First	Group 1-Second	Group 2-Second
M	8:30-11:30	*		*	*	*
	12:00 -3:00					
	3:30-6:30					
T	8:30-11:30		*	*		*
	12:00 -3:00					
	3:30-6:30					
W	8:30-11:30		*	*	*	*
	12:00 -3:00					
	3:30-6:30					
R	8:30-11:30					
	12:00 -3:00					
	3:30-6:30					
F	8:30-11:30		*	*	*	*
	12:00 -3:00					
	3:30-6:30					
S	8:30-11:30	*		*	*	*
	12:00 -3:00					
	3:30-6:30					
M	8:30-11:30		*	*		*
	12:00 -3:00					
	3:30-6:30					
T	8:30-11:30					
	12:00 -3:00					
	3:30-6:30					
W	8:30-11:30	*		*		
	12:00 -3:00					
	3:30-6:30				*	*
R	8:30-11:30	*		*		
	12:00 -3:00					
	3:30-6:30				*	*
F	8:30-11:30		*	*		
	12:00 -3:00					
	3:30-6:30					
S	8:30-11:30					
	12:00 -3:00					
	3:30-6:30					
M	8:30-11:30	*		*		
	12:00 -3:00					
	3:30-6:30					
T	8:30-11:30					
	12:00 -3:00					
	3:30-6:30			*		
W	8:30-11:30		*			
	12:00 -3:00					
	3:30-6:30	*		*		

The result emphasizes that although the runtime decreases considerably, the quality of final output worsens significantly too. The quality of results depends on the sequence of departments in the scheduling phase. This is due to the fact that, each time a department is scheduled, its available rooms capacity depends on the rooms capacity consumed by the departments that have been already scheduled. By using the fourth model approach, it takes 0.03 hours to schedule both departments. It is almost 315 and 40 times lower than the initial and second models respectively and almost 166 times lower than the third model. In the fourth model, the first department is scheduled in a 15 day period, and the second department is scheduled in a 10 day period, which means that all the examinations are scheduled in a 15 day period. This is shorter than the third model but still longer than the first and second models. The reason is the same as the third model. Therefore, the quality of the fourth model decreases in comparison with the first and second models with respect to the length of the scheduled period, but it still provides better results in comparison with the third model. By using the fourth model we are able to take advantage of lower running time in real size datasets, although we cannot get the same advantage in our defined sample.

3.6.5. Model Five – Grouping by Considering Beta-Modified Approach

The fifth model proceeds in a way similar to the fourth model. The only difference is that, in this model, the groups, whose number of members is lower than a pre-defined value, are not merged into one group. Instead, for each of the members of these groups, the best group to merge into, which is the smallest group with the minimum dissimilarity, is found. The groups that do not meet the minimum size requirement are called the Guest Groups. The members of the guest groups are called the Seekers. The groups that accept the incorporation of the seekers are called the Host Groups. The dissimilarity criterion is defined as the number of courses in the course list of the seeker that do not appear in that of the host group. For each of the seekers, the potential host groups are those whose course list covers all the courses registered by the seeker. Among the potential host groups, the best candidate is one that; first, its course list consists of the minimum extra courses in comparison with the seeker's course list, and second, the number of its members is the minimum number among the other potential host groups. If no suitable host group is found, the seeker is put into a new group.



Reg. Info
Divide into N Deps.
Select Dep. # 1
Group Students based on Course Combination
YES
# Members in each Group <α
NO
YES
Select Next Dep.
Remove the Group Look through the other Group from the Smallest to the Largest, to find which group its list of course is the seb-set
NO
Put into the last Group
YES
Any Group Found?
All the Deps. Are Scheduled?
YES
Finish
NO
Put into the last Group
Days = d
Model the ILP
Days = Days + 1
Solve by CPLEX
YES
NO
Feasible Solution ?

Figure 7. Grouping Model by Considering Beta value-Modified approach

Applying this approach, contrary to the third and fourth models, the group with an unreasonably big course list is not formed. So, the problem is expected to solve with higher quality with respect to the examination period length. The pre-defined value, called beta in order to be differentiated from the alpha, represents the maximum size of a guest group. Figure 7 shows the fifth model procedure when considering the beta value. All the remaining steps are the same as the previous model. For each department, students are divided into groups based on their course combination. The number of members in each group is compared with the beta value so that the guest groups, whose number of members is lower than the beta value, are detected. Then, the guest groups are removed, and the corresponding seekers are merged into the best suited host groups based on their closeness. If any seeker cannot be put into the host groups, it is put into a new group.

3.6.5.1. Model Five Input Requirements

All input requirements for model 5 are exactly the same as the model 4 requirements.

3.6.5.2. Model Five Sample

In the previous model, we break our input into two departments. The first one includes the first and second year students, and the next one includes the remaining year students. An important point is that the students of any department cannot cross over the other departments. This presumes that examinations are not repetitive for students, and students are not assigned to an examination twice.

Applying the fifth model with considering beta as equal to 3, for the first department as shown in Table 15, the corresponding groups of the students 2, 4, 6, 7, 11, 15, 19, and 24, who were placed into the groups with lower than 3 members, are removed from the list of groups. After removal of the guest groups, only two groups remain as the host groups; first, the registered for the courses ANAT 1010, COMM 1010, ECED 2000, ECON 1101, ENGI 2200, and HIST 1501, and second, the registered for the courses BIOL 1010, CHEM 1011, CLAS 1100, CSCI 1100, ENGI 1101, and ERTH 1060. The next step is to place the seekers into the best suited host groups. For instance, students 2 and 19, who registered for the courses BIOL 1010, CLAS 1100, CSCI 1100, and ERTH 1060, seek for the best suited host group. Among the host groups, the second host group's course list covers the seekers 2 and 19's course lists, while consisting of two additional courses CHEM 1011 and ENGI

1101. Since the second host group fits the requirements of the seekers 2 and 19 best, the seekers are placed into this group.

Applying the fifth model, considering beta as equal to 3, for the second department as shown in Table 16, the corresponding groups of the students 9, 12, 13, 17, 18, and 20, who were placed into the groups with lower than 3 members, are removed from the list of groups. After removal of the guest groups, only one group remains as the host group, who registered for the courses HIST 1701, MATH 1010, MUSC 1020, PHYC 1100X, STAT 1060, and THEA 1000X. The next step is to find out whether the host group satisfies the seekers requirements or not. For instance, the host group's course list covers seeker 12's course list, who registered for the courses MATH 1010, PHYC 1100X, and THEA 1000X, while consisting of three additional courses HIST 1701, MUSC 1020, and STAT 1060. Since the host group fits the requirements of seeker 12 best, the seeker is placed into this group. As another instance, the host group's course list does not cover the course MATH 1000, which exists in seeker 9's course list. Since the host group does not fit the requirements of seeker 9 completely, the seeker is placed into a new group. Tables 23 and 24 show the members of the groups in the first and second departments, respectively.

Table 23. Members of the Groups of the First Department using the Fifth Model (Considering Beta=3)

Groups	Group 1	Group 2
Number of Members	8	7
Members	0-10-14-21-6-11-15-24	1-8-23-2-19-4-7

Table 24. Members of the Groups of the Second Department using the Fifth Model (Considering Beta=3)

Groups	Group 1	Group 2
Number of Members	6	4
Members	3-5-16-22-12-20	9-17-13-18

Tables 25 and 26 show the list of registered courses associated with the students in each group of the first and second departments, respectively, using the fifth model approach.

Table 25. Course Selection for Each Group of the First Department using the Fifth Model

Group	Course										
	ANAT1010	BIOL1010	CHEM1011	CLAS1100X	COMM1010	CSCI1100	ECON1101	ENGI1101	ENGI2200	ERTH1060	HIST1501
1(0-10-14-21-6-11-15-24)	1				1		1		1		1
2(1-8-23-2-19-4-7)		1	1	1		1		1		1	

Table 26. Course Selection for Each Group of the Second Department using the Fifth Model

Group	Course						
	HIST1701	MATH1000	MATH1010	MUSC1020	PHYC1100X	POLI1020	THEA1000X
1(3-5-16-22-12-20)	1		1	1	1		1
2(9-17-13-18)	1	1	1	1	1	1	1

Table 27. Examinations Schedule using the Fifth Model

			ANAT1010	BIOL1010	CHEM1011	CLAS1100X	COMM1010	CSCI1100	ECED2000	ECON1101	ENGI1101	ENGI2200	ERTH1060	HIST1501	HIST1701	MATH1000	MATH1010	MUSC1020	PHYC1100X	POLI1020	STAT1060	THEA1000X
Gym	M	8:30-11:30		*										*	*							
		12:00 -3:00																				
		3:30-6:30																				
	T	8:30-11:30			*					*						*						
		12:00 -3:00																				
		3:30-6:30																				
	W	8:30-11:30				*	*										*					
		12:00 -3:00																				
		3:30-6:30																				
	R	8:30-11:30																				
		12:00 -3:00																				
		3:30-6:30																				
	F	8:30-11:30						*										*				
		12:00 -3:00																				
		3:30-6:30	*																			
	S	8:30-11:30									*								*			
		12:00 -3:00																				
		3:30-6:30							*													
	M	8:30-11:30																		*		
		12:00 -3:00																				
		3:30-6:30										*	*									
	T	8:30-11:30																				
		12:00 -3:00																				
		3:30-6:30																				
	W	8:30-11:30																				
		12:00 -3:00																				
		3:30-6:30																			*	
	R	8:30-11:30																				
		12:00 -3:00																				
		3:30-6:30																				*

Table 27 shows the optimal examination schedule for the group-course registrations shown in Tables 25 and 26. As shown in Table 27, all the examinations are assigned to the Gym, which is in agreement with the user cost preferences. Table 28 shows the optimal examination schedule for each group at different days and timeslots.

Table 28. Examinations Schedule for each Group using the Fifth Model

		Group 1-First	Group 2-First	Group 1-Second	Group 2-Second
M	8:30-11:30	*	*	*	*
	12:00 -3:00				
	3:30-6:30				
T	8:30-11:30	*	*		*
	12:00 -3:00				
	3:30-6:30				
W	8:30-11:30	*	*	*	*
	12:00 -3:00				
	3:30-6:30				
R	8:30-11:30				
	12:00 -3:00				
	3:30-6:30				
F	8:30-11:30		*	*	*
	12:00 -3:00				
	3:30-6:30	*			
S	8:30-11:30		*	*	*
	12:00 -3:00				
	3:30-6:30	*			
M	8:30-11:30				*
	12:00 -3:00				
	3:30-6:30	*	*		
T	8:30-11:30				
	12:00 -3:00				
	3:30-6:30				
W	8:30-11:30				
	12:00 -3:00				
	3:30-6:30			*	*
R	8:30-11:30				
	12:00 -3:00				
	3:30-6:30			*	*

It is worth mentioning that by applying the fifth model approach, the number of variables decreased in comparison with all the previous models and the runtime decreased comparative to all the previous models. All the variables associated with the original 25 students were substituted with 2 problems each consisting of 2 variables associated with the groups in the first and second departments, as shown in Tables 23 and 24. Although the decrease in the number of variables seems to be considerable for the sample problem in comparison with first model, it will more significantly affect the number of variables in the large-size problems, which deal with real datasets.

The result shows that the output quality decreases in comparison with the first and second models, but increases in comparison with the third and fourth model. The result shows that the runtime decreases dramatically in comparison with the first, second, and third models, but remains almost unchanged in comparison with the fourth model. Although the output quality and runtime have almost improved in comparison with the fourth model, they are expected to show significant improvements for the large size problems dealing with real datasets. Similar to the fourth model, the quality of results obtained by the fifth model also depends on the sequence of departments in the scheduling phase. This is due to the fact that each time a department is scheduled, its available rooms capacities depends on the rooms capacities consumed by the departments that have been already scheduled. By using this new approach, the solution computed in 0.002 hours to schedule both departments. It is almost 4750 and 600 times faster than the initial and second models respectively and 2500 times faster than the third model, and 15 times faster than fourth model. In the fifth model, the first department is scheduled in a 10 day period, and the second department is scheduled in a 7 day period, which means that all the examinations are scheduled in a 10 day period. Applying the fifth model, the examination period length decreases by 30% in comparison with the fourth model. The reason is that in the fifth model, contrary to the fourth model, the groups with unreasonably large course lists are not formed. The groups with larger course lists force the model to schedule the timetable in a longer period due to the presence of the seventh constraint, which emphasized having one day off after three consecutive examinations. In the absence of such groups, the problem is solved with higher quality with respect to the examination period length.

3.6.6. Models Six and Seven – Modified Approach based on Models Two and Five

Models 6 and 7 originate from models 2 and 5, respectively. The only difference is that they take the cost of exam period length into account. By adding the cost of exam period length (*pw*) into the objective function, the models are forced to decrease the exam period length (*w*). To force the models to schedule the exams in a smaller period length, constraint 8 is added to the model, which reflects the relation between scheduled exam times ($t * x_{srtc}$) and the exam period length (*w*). Since taking the exams in a shorter length is preferred to taking them in cheap rooms, the cost of exam period length is set at a significantly higher level in comparison with the rooms costs. Applying this approach, we can take the advantage of high quality solutions and shorter exam period length at the same time.

3.6.6.1. Models Six and Seven Input Requirements

All input requirement for models 6 and 7 are exactly the same as for models 2 and 5, respectively. Besides, we add the cost of exam period length, equal to 1000 for our small sample.

3.6.6.2. Models Six and Seven Sample Data

Using the sixth model, Table 29 shows the optimal examination schedule for the group-course registrations shown in Table 8.

Table 29. Examinations Schedule using the Sixth Model

			ANAT1010	BIOL1010	CHEM1011	CLAS1100X	COMM1010	CSCI1100	ECED2000	ECON1101	ENGL1101	ENGL2200	ERTH1060	HIST1501	HIST1701	MATH1000	MATH1010	MUSC1020	PHYC1100X	POLI1020	STAT1060	THEA1000X
Gym	M	8:30-11:30		*			*												*			
		12:00 -3:00																				
		3:30-6:30																				
	T	8:30-11:30				*			*									*				
		12:00 -3:00																				
		3:30-6:30																				
	W	8:30-11:30						*		*										*		*
		12:00 -3:00																				
		3:30-6:30																				
	R	8:30-11:30																				
		12:00 -3:00																				
		3:30-6:30																				
	F	8:30-11:30										*	*									
		12:00 -3:00																				
		3:30-6:30															*					
	S	8:30-11:30			*									*								
		12:00 -3:00																				
		3:30-6:30													*							
	M	8:30-11:30																				
		12:00 -3:00																				
		3:30-6:30	*								*					*					*	

By applying the sixth model, the number of variables decreased in comparison with models one and three, and the runtime decreased compared to models one and three. The runtime of this model is better compare to model one but has no improvement in comparison to the second model. In the second model, the examinations are scheduled in a 7 day period.

Using the seventh model, Table 30 shows the optimal examination schedule for the group-course registrations of the first and second departments shown in Tables 25 and 26, respectively.

Table 30. Examinations Schedule using the Seventh Model

			ANAT1010	BIOL1010	CHEM1011	CLAS1100X	COMM1010	CSCI1100	ECED2000	ECON1101	ENGI1101	ENGI2200	ERTH1060	HIST1501	HIST1701	MATH1000	MATH1010	MUSC1020	PHYC1100X	POLI1020	STAT1060	THEA1000X
Gym	M	8:30-11:30		*										*	*							
		12:00 -3:00																				
		3:30-6:30																				
	T	8:30-11:30			*					*						*						
		12:00 -3:00																				
		3:30-6:30																				
	W	8:30-11:30				*	*										*					
		12:00 -3:00																				
		3:30-6:30																				
	R	8:30-11:30																				
		12:00 -3:00																				
		3:30-6:30																				
	F	8:30-11:30						*										*				
		12:00 -3:00																				
		3:30-6:30	*																			
	S	8:30-11:30									*								*			
		12:00 -3:00																				
		3:30-6:30							*													
	M	8:30-11:30																		*		
		12:00 -3:00																				
		3:30-6:30										*	*									
	T	8:30-11:30																				
		12:00 -3:00																				
		3:30-6:30																				
	W	8:30-11:30																				
		12:00 -3:00																				
		3:30-6:30																			*	
	R	8:30-11:30																				
		12:00 -3:00																				
		3:30-6:30																				*

By applying the seventh model, the number of variables decreased in comparison with the previous models, and the runtime decreased compared to all previous models except model 5. The result shows that the output quality with respect to the exam period length decreases in comparison with the third and fourth models, increases in comparison with the first, second and sixth models and remains constant in comparison with the fifth model. The result shows that the runtime decreases dramatically in comparison with the first, second, third, fourth and sixth models, while increasing slightly in comparison with the fifth model. Although the output quality and runtime have almost improved in comparison with other models, they are expected to show significant improvement for the large size problems dealing with real datasets. By using the new approach, the solver computed for 0.008 hours to schedule both departments. It is almost 149 times faster than the second model, 625 times faster than the third model, and almost 4 times faster than the fourth model. As with the fifth model, the examinations are scheduled in a 10 day period. In the real datasets, the last model, in comparison with the fourth model, is expected to provide results with higher quality with respect to the length of the scheduled period. Using the seventh model, Table 31 shows the optimal examination schedule for each group at different days and timeslots.

3.7. Model Summary

The Figure 8 presents a summary of the models. The arrows represent model inheritances.

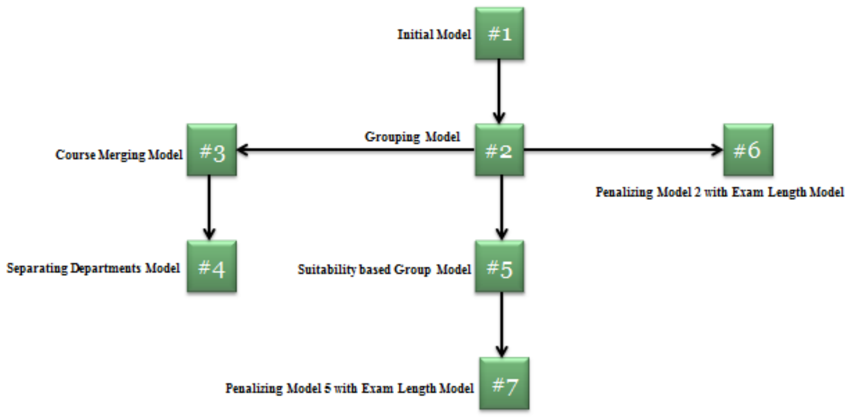

Initial Model
#1
Grouping Model
#2
Course Merging Model
#3
#6
Penalizing Model 2 with Exam Length Model
Separating Departments Model
#4
Suitability based Group Model
#5
Penalizing Model 5 with Exam Length Model
#7

Figure 8. Summary of the Models

3.8. Sensitivity Analysis

The performances of models three and four depend on the pre-defined alpha value, and the performances of models five and seven depend on the pre-defined beta value. We investigate 3 parameters settings for both the alpha and beta values. Table 32 shows the indicators of Model Performance based on different parameters and models. Sensitivity analysis shows that in models three and four, an increase in the alpha value dramatically decreases the run-time due to the fact that the number of groups decreases. In model three, an increase in the alpha value results in an increase in the exam period length due to the presence of one group with too many courses, while in model four, it results in a decrease in the exam period length since such a group is not formed. Sensitivity analysis on the parameters of the models five and seven shows that none of the models' performance indicators, run-time and exam period length, are sensitive to changes in beta value.

Table 31. Examinations Schedule for each Group using the Seventh Model

		Group 1-First	Group 2-First	Group 1-Second	Group 2-Second
M	8:30-11:30	*	*	*	*
	12:00 -3:00				
	3:30-6:30				
T	8:30-11:30	*	*		*
	12:00 -3:00				
	3:30-6:30				
W	8:30-11:30	*	*	*	*
	12:00 -3:00				
	3:30-6:30				
R	8:30-11:30				
	12:00 -3:00				
	3:30-6:30				
F	8:30-11:30		*	*	*
	12:00 -3:00				
	3:30-6:30	*			
S	8:30-11:30		*	*	*
	12:00 -3:00				
	3:30-6:30	*			
M	8:30-11:30				*
	12:00 -3:00				
	3:30-6:30	*	*		
T	8:30-11:30				
	12:00 -3:00				
	3:30-6:30				
W	8:30-11:30				
	12:00 -3:00				
	3:30-6:30			*	*
R	8:30-11:30				
	12:00 -3:00				
	3:30-6:30			*	*

Table 32. Sensitivity Analysis

	Run-Time				Exam Period Length			
Alpha-Beta\ Models	Model 3	Model 4	Model 5	Model 7	Model3	Model 4	Model 5	Model 7
2	28	0.03	0.003	0.009	23	15	10	10
3	9.5	0.03	0.002	0.008	26	15	10	10
4	2	0.003	0.003	0.009	26	10	10	10

3.9. Comparing of the Results

Table 33 shows a comparison between the number of days, number of variables, number of constraints, and runtime of different models based on the same input. Each row corresponds to one model. The number of days, number of variables, number of constraints and run-time of each model are respectively shown in the first, second, third and fourth columns.

Table 33. Results Comparison

Number of Models	Number of Days in exam period	Number of Variables	Number of Constraints	Run-time (hours)
Model 1	**7**[+]	**31500**	**24324**	**9.5**
Model 2	**7**[+]	**16380**	**5151**	**1.19**
Model 3	**26**	**18720**	**5430**	**5**
Model 4- First Dept.	**15**	**3240**	**1225**	**0.03**
Model 4- Second Dept.	**10**	**1440**	**808**	
Model 5-First Dept.	**7**	**1512**	**199**[^]	**0.002**[×]
Model5- Second Dept.	**10**	**1440**[*]	**808**	
Model 6	**7**[+]	**16381**	**21531**	**1.97**
Model 7-First Dept.	**7**	**1513**	**1711**	**0.008**
Model 7-Second Dept.	**10**	**1441**	**2248**	

[+] *Minimum number of days*

[*] *Minimum number of variables*

[^] *Minimum number of constraints*

[×] *Minimum run-time*

It shows that the run-time, as expected, decreases as the number of variables and constraints decreases, except for models 6 and 7 because of applying on additional constraint. It is shown that the first and second models are scheduled in the minimum total number of days although the number of variables, the number of constraints and the run-time of the first model is much higher than other models, especially model 7. The third model requires more days for examination scheduling in comparison with other models. The reason is that the group, which is built as a combination of the groups with fewer than alpha members, has a considerably high number of courses to be scheduled simultaneously. This forces the third model, in regards to the seventh constraint, to schedule the examinations in a longer period of time. Its number of variables, number of constraints and run-time significantly decreases compared to the first model, while it does not significantly change compared to the second model. However, this model has lower running time in real-size datasets compared to the first and second models. The fourth and fifth models run in significantly lower times. Our tests show that in real-size datasets, the fifth model significantly outperforms the fourth model. In the seventh model, the run-time decreases considerably compared to the first, second, third and fourth models, while the total number of days required for examination scheduling for both departments increases compared to models one and two. The quality of the results for models four, five, and seven depends on the sequence of the departments in the scheduling phase.

It can be concluded that models 5 and 7 are preferred in the case of real-size datasets due to low run-time and near optimal solutions with respect to exam period length. Among all the models, the fifth model runs significantly faster, although the number of days required for examination scheduling is longer than the first and second models. If we have no limitation on running time, the second model is the best model with regards to exam scheduling length.

3.10. Model Performance

Figure 9 shows the trend of improvements in the proposed models. As shown in Figure 9, the first model produces the shortest exam period length, meanwhile, disadvantage of an extremely high run-time. In order to improve the run-time, we presented the second model as a grouping approach. By applying this model, the number of variables and constraints decreased, thereby giving the advantage of shorter run-time while keeping the exam period length constant. In order to improve this model with respect to the run-time, we introduced the third model. In this model, we defined grouping by considering an alpha value, in order to merge the groups with lower than alpha members into one group. Due to the fact that we have one group with too many courses, against our expectation, the exam period length and run-time increase considerably. In the fourth model, by dividing the problem into two smaller problems, we decreased the size of problem and got the advantage of shorter run-time, but still met the difficulty of long exam period length. In order to improve the exam period length, we defined the fifth model. In order to solve the difficulty of forming one group with too many courses, we presented a grouping approach to find the best suited group for merging each group with less than beta members. By applying this method, we got the advantage of the lowest run-time, and close to the best exam period length.

According to the Table 34, if no limitation on the run-time exists, the second model is the best model with respect to the exam period length. So, based on the second model, the sixth model was defined, which takes the exam period length cost into account in order to decrease the exam period length for real datasets. However, it gave the same quality result at the tested level. Furthermore, Table 34 shows that the fifth model provides a close to optimal solution in an extremely short run-time. So, based on the fifth model, the seventh model was defined which takes the exam period length cost into account in order to decrease the exam period length for real datasets; however, it gave the same quality result at the tested level.

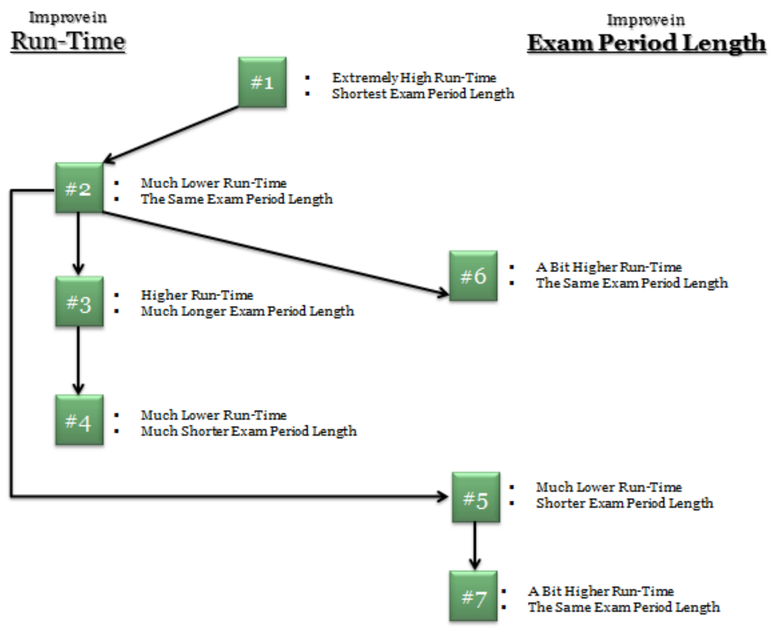


Figure 9. Computational Performance of the Models

CHAPTER 4 : CONCLUSION

In this research, we introduced 7 methods to schedule modified examination-scheduling problems by formulating linear programming equations to cover all the required constraints. The aim of the methods is to assign a certain number of examinations, associated with the same number of courses, to limited available resources such as available space and time provided by the examiner. We developed the models based on the assignment problem. The models introduced are capable of reducing the number of conflicts in the scheduling process. Integer linear programming models, subject to restrictive constraints such as time constraints and rooms capacity, were developed by satisfying all the constraints, a feasible solution is achieved. We defined the following constraints:

- Rooms capacity constraint;
- Time constraints;
- Each student is not allowed to have more than one examination on each day;
- Rooms are available from 8:30 to 11:30, 12:00 to 3:00, and 3:30 to 6:30;
- For each student, the time between the start of examinations should be at least 24 hours;
- Students should not have examinations on more than 3 consecutive days;

In our study, the model was coded in python interfaced with CPLEX.

We introduced 7 different models and compared their results in order to find the qualified output for scheduling problems. The first model scheduled all the courses in a perfect period although its run-time is comparatively higher than all the other models. The second model, the grouping model, is a modification of the first model based on grouping the students who enrol in the same courses. By applying this approach we decreased the run-time in comparison to the initial model while keeping the examination period constant. In the third model, the alpha parameter was defined, which represents the minimum number of students required to build a group, so that small groups are combined in order to decrease the size of the problem. By applying this approach, the model ran two times faster than the initial model but included a group with an unreasonably large member of courses to schedule, which resulted in a decrease in quality of the solution. In the fourth model, which was based on breaking into departments, we divided the students based on their corresponding departments. The important point to be considered is that the students of one department could not cross over other into department courses, by applying this approach, we got the advantage of lower run-time compared to all previous models. Therefore, we still have a group with an unreasonably large number of courses to be scheduled. In order to overcome this problem, we introduced the fifth model. In this model, contrary to the third model, the groups with lower than alpha members are not merged into one group, but they are assigned to the most similar and smallest groups. Thereby, we could get considerable advantage of shorter run-time while still keeping the high quality of output with respect to the number of days required as the examination period. Models 6 and 7 are generated from models 2 and 5 respectively by taking the cost of exam period length into account in order to decrease the exam period. By applying this approach for model 7, we can take advantage of a high quality solution and shorter exam period length at the same time in comparison to models three and four. Meanwhile, by applying this method for model 6, we can get a solution close to the one provided by model two.

Our results showed that the second model provided good results with respect to exam period length. But the fifth and seventh models can provide results in a reasonable run-time with the cost of insignificant loss in the quality of results.

Reference List

[1] S. Abdennadher, M. Aly and M. Edward. Constraint-based timetabling system for the german university in cairo. Applications of Declarative Programming and Knowledge Management 5437pp. 69-81. 2009.

[2] H. Asmuni, E. K. Burke and J. M. Garibaldi. A comparison of fuzzy and non-fuzzy ordering heuristics for examination timetabling. Proceedings of the 5th International Conference on Recent Advances in Soft Computing (RASC 2004) 2004.

[3] H. Asmuni, E. Burke, J. Garibaldi and B. McCollum. Fuzzy multiple heuristic orderings for examination timetabling. PATAT'04 Proceedings of the 5th International Conference on Practice and Theory of Automated Timetabling 3616pp. 334-353. 2005.

[4] M. Ayob and G. Kendall, "A Monte Carlo Hyper-Heuristic To Optimise Component Placement Sequencing ForMulti Head Placement Machine," PLACEMENT MACHINE, INTECH'03 THAILAND, pp. 132-141, 2003.

[5] M. Ayob, S. Abdullah and A. Malik, "A Practical Examination Timetabling Problem at the Universiti Kebangsaan Malaysia," vol. 7, 2007.

[6] 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 and N. R. Sabar. Intelligent examination timetabling software. Procedia - Social and Behavioral ScienceS 18pp. 600-608. 2011.

[7] M. Ayob, A. Malik, S. Abdullah, A. Hamdan, G. Kendall and R. Qu. Solving a practical examination timetabling problem: A case study. Proceedings of the 2007 International Conference on Computational Science and its Applications 4707pp. 611-624. 2007.

[8] P. Boizumault, Y. Delon and L. Peridy. Constraint logic programming for examination timetabling. The Journal of Logic Programming 26(2), pp. 217-233. 1996.

[9] P. Boizumault, C. Guéret and N. Jussien, "Efficient Labelling and Constraint Relaxation for Solving Time Tabling Problems," Proceedings of the 1994 ILPS Post-Conference Workshop on Constraint Languages/Systems and their use in Problem Modelling, pp. 116-130, 1994.

[10] E. K. Burke, Y. Bykov, J. P. Newall and S. Petrovic, "A Time-predefined approach to course timetabling," Yugoslav Journal of Operations Research, vol. 13, pp. 139-151, 2003.

[11] E. Burke, J. Newall and R. Weare. A memetic algorithm for university exam timetabling. Selected Papers from the First International Conference on Practice and Theory of Automated Timetabling 1153pp. 241-250. 1996.

[12] E. K. Burke and J. P. Newall. A multistage evolutionary algorithm for the timetable problem. Evolutionary Computation, IEEE Transactions on 3(1), pp. 63-74. 1999.

[13] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic and R. Qu. A graph-based hyper-heuristic for educational timetabling problems. European Journal of Operational Research 176(1), pp. 177-192. 2007.

[14] E. K. Burke and S. Petrovic. Recent research directions in automated timetabling. European Journal of Operational Research 140(2), pp. 266-280. 2002.

[15] E. Burke and Y. Bykov, "Solving exam timetabling problem with the Flex-Deluge Algorithm ," CONFERENCE PROCEEDINGS, pp. 370-372, 2006.

[16] E. Burke, D. Elliman and R. Weare, "A Genetic Algorithm Based University Timetabling System," Proceedings of the 2nd East-West International Conference on Computer Technologies in Education, pp. 35-40, 1994.

[17] E. Burke, G. Kendall, M. Mısır and E. Özcan. Monte carlo hyper-heuristics for examination timetabling. Annals of Operations Research 196(1), pp. 73-90. 2012.

[18] M. Caramia, P. Dell'Olmo and G. Italiano. New algorithms for examination timetabling. WAE '00 Proceedings of the 4th International Workshop on Algorithm Engineering 1982pp. 230-242. 2001.

[19] M. Caramia, P. Dell'Olmo and G. Italiano. New algorithms for examination timetabling. Proceedings of the 4th International Workshop on Algorithm 1982pp. 230-242. 2001.

[20] M. W. Carter, G. Laporte and J. W. Chinneck. A general examination scheduling system. INFORMS 24(3), pp. pp. 109-120. 1994.

[21] M. W. Carter, G. Laporte and S. Y. Lee. Examination timetabling: Algorithmic strategies and applications. European Journal of Operational Research Society 47(3), pp. pp. 373-383. 1996.

[22] S. Casey and J. Thompson. GRASPing the examination scheduling problem. Computer Science 2740pp. 232-244. 2003.

[23] S. L. Chan and Y. Zhang. EMS: An examination scheduling and management system. Expert Systems with Applications 12(3), pp. 311-321. 1997.

[24] P. Cote, T. Wong and R. Sabourin, "Application of a hybrid multi-objective evolutionary algorithm to the uncapacitated exam proximity problem," Proceedings of the 5th International Conference on Practice and Theory of Automated Timetabling, vol. 3616, pp. 294-312, 2004.

[25] S. Daskalaki and T. Birbas. Efficient solutions for a university timetabling problem through integer programming. European Journal of Operational Researc 160(1), pp. 106-120. 2005.

[26] P. David. A constraint-based approach for examination timetabling using local repair techniques. PATAT '97 Selected Papers from the Second International Conference on Practice and Theory of Automated Timetabling II 1408pp. 169-186. 1998.

[27] L. Di Gaspero and A. Schaerf. Tabu search techniques for examination timetabling. PATAT '00 Selected Papers from the Third International Conference on Practice and Theory of Automated Timetabling III 2079pp. 104-117. 2001.

[28] G. Dueck. New optimization heuristics: The great deluge algorithm and the record-to-record travel. Journal of Computational Physics 104(1), pp. 86-92. 1993.

[29] T. Duong and K. Lam, "Combining Constraint Programming and Simulated Annealing on University Exam Timetabling," 2nd International Conference, pp. 205-210, 2004.

[30] E. Ozcan, "Final exam scheduler," Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1356-1363, 2005.

[31] M. Eley. Ant algorithms for the exam timetabling problem. PATAT'06 Proceedings of the 6th International Conference on Practice and Theory of Automated Timetabling VI 3867pp. 364-382. 2007.

[32] W. Erben. A grouping genetic algorithm for graph colouring and exam timetabling. PATAT '00 Selected Papers from the Third International Conference on Practice and Theory of Automated Timetabling III 2079pp. 132-156. 2001.

[33] A. Ergül. GA-based examination scheduling experience at middle east technical university. First First International Conference on Practice and Theory of Automated Timetabling 1153pp. 212-226. 1996.

[34] E. Falkenauer, "Genetic Algorithms and Grouping Problems," Univ. Libre De Bruxelles and Optimal Design Belgium, pp. 1-238, 1998.

[35] M. Fathian, B. Amiri and A. Maroosi. Application of honey-bee mating optimization algorithm on clustering. Applied Mathematics and Computation 190(2), pp. 1502-1513. 2007.

[36] 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 194(1), pp. 203-221. 2012.

[37] T.B, Gibel. An operating room scheduling problem solved using Simulated Annealing.2007. Dalhousie University

[38] J. Joshua Thomas and A. Tajudin, "Visualizing the Examination Timetabling Data Using Clustering Method and TreeMaps," Proceedings of the 2nd IMT-GT Regional Conference on Mathematics, Statistics and Application University Sains Malaysia, pp. 454-458, 2006.

[39] G. Kendall and N. Hussin. An investigation of a tabu-search-based hyper-heuristic for examination timetabling. 1st International Conference, MISTA '03 Nottingham, UK, pp. 309-328. 2005.

[40]S.l. Kripakaran. Integer Programming approach to university space planning. 2007. Dalhousie University.

[41]E. Lawler. Combinatorial Optimization Networks and Matroids. 1976. 0-486-41453-1.

[42] B. Paechter, A. Cumming, H. Luchian and M. Petriuc. Two solutions to the general timetable problem using evolutionary methods. Presented at Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on. 1994, .

[43] L. F. Paquete and C. M. Fonseca, "A Study of Examination Timetabling with Multiobjective Evolutionary Algorithms," Proceedings of the 4th Metaheuristics International Conference (MIC 2001), pp. 149-154, 2001.

[44] N. Pillay. An informed genetic algorithm for the examination timetabling problem. Journal Applied Soft Computing 10(2), pp. 457-467. 2010.

[45] P. Rattadilok, A. Gaw and R. Kwan. Distributed choice function hyper-heuristics for timetabling and scheduling. 3616pp. 51-67. 2005.

[46] P. Ross, E. Hart and D. Corne. Some observations about GA-based exam timetabling. Computer Science 1408pp. 115-129. 1998.

[47] N. R. Sabar and M. Ayob. Examination timetabling using scatter search hyper-heuristic. Presented at Data Mining and Optimization, 2009. DMO '09. 2nd Conference on. 2009, .

[48] N. R. Sabar, M. Ayob and G. Kendall. Tabu exponential monte-carlo with counter heuristic for examination timetabling. Presented at Computational Intelligence in Scheduling, 2009. CI-Sched '09. IEEE Symposium on. 2009, .

[49] N. Sabar, M. Ayob, G. Kendall and R. Qu. Roulette wheel graph colouring for solving examination timetabling problems. Annual International Conference on Combinatorial Optimization and Applications 5573pp. 463-470. 2009.

[50] S. Shimoji and S. Lee. Data clustering with entropical scheduling. Presented at Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on. 1994 .

[51] Tang Van To and Soe San Win. Clustering approach to examination scheduling. 2010,

[52] J. Thompson and K. Dowsland. Variants of simulated annealing for the examination timetabling problem. Annals of Operations Researc 63(1), pp. 105-128. 1996.

[53] J. Thompson and K. Dowsland. Variants of simulated annealing for the examination timetabling problem. Annals of Operations Research 63(1), pp. 105-128. 1996.

[54] H. Turabieh and S. Abdullah. An integrated hybrid approach to the examination timetabling problem. The International Journal of Management Science 39(6), pp. 598-607. 2011.

[55] R. Weare, E. Burke and D. Elliman, "A Hybrid Genetic Algorithm for Highly Constrained Timetabling Problems," Proceedings of the 6th International Conference on Genetic Algorithms, pp. 605-610, 1995.

[56] G. M. White. Using tabu search with longer-term memory and relaxation to create examination timetables. European Journal of Operational Research 153(1), pp. 80-91. 2004.

[57] T. Wong, P. Cote and P. Gely. Final exam timetabling: A practical approach. Presented at Electrical and Computer Engineering, 2002. IEEE CCECE 2002. Canadian Conference on. 2002,

Appendix I. Code for Model Seven

The following steps elaborate our Initial Python code. By adding few steps we develop new models in order to improve our final solution.

```
import time
import xlrd, xlwt
from numpy import math
import cplex
from cplex.exceptions import CplexError
import shlex
from sys import argv
file = str(argv[1])

class Problem(object):
    def __init__(self, objective, var_names, var_types, lower_bnds, upper_bnds,
sense, rhs, fraction):
        self.obj      = objective
        self.names    = var_names
        self.types    = var_types
        self.lb       = lower_bnds
        self.ub       = upper_bnds
        self.sense    = sense
        self.rhs      = rhs
        self.fraction = fraction

start = time.time()

#Read from excel file
wb1 = xlrd.open_workbook('input/%s.xls' %file)
wb2 = xlwt.Workbook()
capacity_sheet = wb1.sheet_by_name('Capacity')
cost_sheet     = wb1.sheet_by_name('Cost')
course_sheet   = wb1.sheet_by_name('Course')
room_sheet     = wb1.sheet_by_name('Room')
Time_sheet     = wb1.sheet_by_name('Time')
day_sheet      = wb1.sheet_by_name('Day')
alpha_sheet    = wb1.sheet_by_name('alpha')

alpha = int(alpha_sheet.cell(0,0).value)
room  = int(room_sheet.nrows)
Time  = int(Time_sheet.nrows)
day   = int(day_sheet.cell(0,0).value)

day_list  =["Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"]

room_list =[]
for i in range (room):
    room_list.append(room_sheet.cell(i,0).value)

Time_list =[]
for i in range (Time):
    Time_list.append(Time_sheet.cell(i,0).value)

capacity = [["" for x in range (10*Time*day)] for y in range (room)]
adjacent = [["" for x in range (10*Time*day)] for y in range (room)]
for i in range (room):
    for j in range(10*Time*day):
        capacity[i][j] = capacity_sheet.cell(i,0).value
        adjacent[i][j] = capacity_sheet.cell(i,1).value
```

```
cost = []
for i in range (room):
    cost.append(cost_sheet.row_values(i))

wb2.add_sheet('result-course')
for i in range (room):
    for j in range (10*day):
        for k in range (Time):
            wb2.get_sheet(0).write(10*day*Time*i    +    Time*j    +    k    +
1,0,room_list[i])
            wb2.get_sheet(0).write(10*day*Time*i    +    Time*j    +    k    +
1,1,day_list[j%6])
            wb2.get_sheet(0).write(10*day*Time*i    +    Time*j    +    k    +
1,2,Time_list[k])

wb2.add_sheet('result-group')
for j in range (10*day):
    for k in range (Time):
        wb2.get_sheet(1).write(Time*j + k + 1,0,day_list[j%6])
        wb2.get_sheet(1).write(Time*j + k + 1,1,Time_list[k])

################################################################################

def one_by_one_solver():

    department_sheet = wb1.sheet_by_name('Departments')
    department = int(department_sheet.nrows)
    department_list = []
    for i in range (department):
        department_list.append(department_sheet.cell(i,0).value)

    previous_course = 0
    previous_group  = 0
    for d in range (department):
        days = int(day_sheet.cell(0,0).value)
        registration_sheet = wb1.sheet_by_name(str(department_list[d]))
        print   "<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<   DEPARTMENT   :   %s
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>" %str(department_list[d])
        print ""

        wb2.add_sheet('members-'+str(department_list[d]))
        wb2.add_sheet('registration-'+str(department_list[d]))

        student = int(registration_sheet.nrows)
        course  = int(registration_sheet.ncols)

        course_list = []
        for j in range (course):
            course_list.append(course_sheet.cell(d,j).value)

        student_registration = list()
        for i in range (student):
            student_registration.append(registration_sheet.row_values(i))
            for j in range (course):
                if student_registration[i][j] == "":
                    student_registration[i][j] = 0.0

        common = [["" for x in range (student)] for y in range (student)]
        for i in range (student):
            counter = 0
            for j in range(student):
                if student_registration[i] == student_registration[j]:
```

```
            common[i][counter] = j
            counter += 1

group_members = list()
for i in range (student):
    if common[i] not in group_members:
        group_members.append(common[i])
group = len(group_members)

print "******************* Members of the Groups *******************"
print "******************** Initial Grouping    ********************"
print ""
for i in range (len(group_members)):
    print "Group %s:" %(i+1)
    gm = list()
    for j in range (student):
        if group_members[i][j] != "":
            gm.append(group_members[i][j])
    print gm
    print ""

initial_members  = [0 for x in range (group)]
a_group  = list()
a_member = list()
for i in range (group):
    for j in range (student):
        if group_members[i][j] != "":
            initial_members[i] += 1
    if initial_members[i] <= alpha:
        a_group.append(i)
        for a in range (initial_members[i]):
            if group_members[i][a] not in a_member:
                a_member.append(group_members[i][a])

members = initial_members[:]
a_group.reverse()
for i in range (len(a_group)):
    del [group_members[a_group[i]]]
    del [members[a_group[i]]]
group = len(group_members)

print "******************** Members of the Groups **************"
print "***   Removal of the Groups with Less than Alpha Members    ***"
print ""
for i in range (len(group_members)):
    print "Group %s:" %(i+1)
    gm = list()
    for j in range (student):
        if group_members[i][j] != "":
            gm.append(group_members[i][j])
    print gm
    print ""

sorted_members = sorted(members)

rank  = ["" for x in range(len(members))]
for a in range (len(members)):
    for b in range (len(members)):
        if (members[a] == sorted_members[b]) and  (b not in rank):
            rank[a] = b

sequence = ["" for x in range(len(members))]
for a in range (len(members)):
```

```
            for b in range (len(members)):
                if rank[b] == a:
                    sequence[a] = b

        cnt = [0 for x in range (len(members))]
        a_member_modified = a_member[:]
        for a in range (len(a_member)):
            for b in range (len(members)):
                crs        = 0
                common_crs = 0
                for c in range (course):
                    if student_registration[a_member[a]][c] == 1:
                        crs += 1
                        if
student_registration[group_members[sequence[b]][0]][c] == 1:
                            common_crs += 1
                if common_crs == crs:
                    group_members[sequence[b]][members[sequence[b]] + cnt[b]] =
a_member[a]
                    a_member_modified.remove(a_member[a])
                    cnt[b] += 1
                    break

        for i in range (group):
            members[i] = 0
            for j in range (student):
                if group_members[i][j] != "":
                    members[i] += 1

        print "****************** Members of the Groups ********************"
        print "*******************    Final Grouping     *******************"
        print ""
        for i in range (len(group_members)):
            print "Group %s:" %(i+1)
            gm = list()
            for j in range (student):
                if group_members[i][j] != "":
                    gm.append(group_members[i][j])
            print gm
            print ""

        if len(a_member_modified) > 0:
            group += 1
            print "Group %s:" %(group)
            print a_member_modified
            print ""

        print "*************************************************************"

        wb2.get_sheet(2*d+2).write(0,0,"Groups")
        wb2.get_sheet(2*d+2).write(1,0,"# Members")
        group_modified     =  [[""  for  x  in  range  (student)]  for  y  in  range
(group)]
        members_modified = ["" for x in range (group)]
        for i in range (group-1):
            members_modified[i] = members[i]
            wb2.get_sheet(1).write(0,i+2+previous_group,"Group     "+str(i+1)+"-
"+str(department_list[d]))
            wb2.get_sheet(2*d+2).write(0,i+1,"Group "+str(i+1))
            wb2.get_sheet(2*d+2).write(1,i+1,members_modified[i])
            for j in range (student):
                if group_members[i][j] != "":
                    group_modified[i][j] = group_members[i][j]
                    wb2.get_sheet(2*d+2).write(j+2,i+1,group_modified[i][j])

        members_modified[group-1] = len(a_member_modified)
        wb2.get_sheet(1).write(0,group+1+previous_group,"Group   "+str(group)+"-
"+str(department_list[d]))
        wb2.get_sheet(2*d+2).write(0,group,"Group "+str(group))
        wb2.get_sheet(2*d+2).write(1,group,members_modified[group-1])
        for j in range (members_modified[group-1]):
            group_modified[group-1][j] = a_member_modified[j]
            wb2.get_sheet(2*d+2).write(j+2,group,group_modified[group-1][j])

        course_list_combined = list()
        group_registration  =  [[""  for  x  in  range  (group)]  for  y  in  range
(course)]
        for x in range (group):
            wb2.get_sheet(2*d+3).write(x+1,0,"Group "+str(x+1))
        for i in range (course):
            wb2.get_sheet(2*d+3).write(0,i+1,course_list[i])
            for j in range (group-1):
                group_registration[i][j]                                            =
student_registration[group_members[j][0]][i]
                wb2.get_sheet(2*d+3).write(j+1,i+1,group_registration[i][j])
            for s in range (len(a_member_modified)):
                if  (student_registration[a_member_modified[s]][i]==1)  and  (i
not in course_list_combined):
                    course_list_combined.append(i)
        for i in range (course):
            if i in course_list_combined:
                group_registration[i][group-1] = 1

wb2.get_sheet(2*d+3).write(group,i+1,group_registration[i][group-1])
            else:
                group_registration[i][group-1] = 0

wb2.get_sheet(2*d+3).write(group,i+1,group_registration[i][group-1])

        reg = [["" for x in range (group)] for y in range (course)]
        for i in range (course):
            for j in range (group):
                if group_registration[i][j] == 1:
                    reg[i][j] = j
                else:
                    group_registration[i][j] = 0

        combination_2_of_group = [0 for x in range (course)]
        for c in range (course):
            if sum(group_registration[c]) > 1:
                combination_2_of_group[c]                                           =
(math.factorial(sum(group_registration[c])))/(2*math.factorial(sum(group_regist
ration[c])-2))

        if group > 1:
            combination_2_of_all_group                                              =
(math.factorial(group))/(2*math.factorial(group-2))

            combination = [["" for x in range (combination_2_of_all_group)] for
y in range (course)]
            n_constraints = 0
            for i in range(course):
                counter = 0
                if sum(group_registration[i]) > 1:
                    for j in range(group):
                        for k in range(j+1,group):
                            if (reg[i][j] != "") and (reg[i][k] != ""):
                                combination[i][counter] = reg[i][j],reg[i][k]
                                counter += 1
                                n_constraints  += 1

        else:
            combination = [0 for x in range (course)]
            n_constraints = 0

        result                                                                 =
solver(course,course_list,previous_course,group,group_registration,previous_gro
up,combination,combination_2_of_group,members_modified,n_constraints,days)
        while result == False:
            days += 1
            result                                                             =
solver(course,course_list,previous_course,group,group_registration,previous_gro
up,combination,combination_2_of_group,members_modified,n_constraints,days)

        previous_course += course
        previous_group  += group
        print ""
        print "---------------------- UPDATED CAPACITY -----------------------"
        print ""
        for i in range (room):
            print room_list[i], capacity[i]
        print ""

    wb2.save('output/%s.xls' %file)

    run_time = (time.time() - start)/3600
    print "-----------------------------------------------------------------"
    print ""
    print "RUN TIME = ", run_time

    return

###############################################################################

def
solver(course,course_list,previous_course,group,group_registration,previous_gro
up,combination,combination_2_of_group,members_modified,n_constraints,days):

    global names

    print "NUMBER OF DAYS : ", days
    print ""

    try:
        # create empty problem
        my_prob = cplex.Cplex()
        # initialize it
        handle                                                                 =
populate(my_prob,course,group,group_registration,combination,combination_2_of_g
roup,members_modified,n_constraints,days)
        # solve the problem
        my_prob.solve()

        #get number of columns in the solution
        num_columns  = my_prob.variables.get_num()
        #get the solution
        solution = my_prob.solution

        # print minimized objective value
        print "Objective value = " , solution.get_objective_value()
        # get the first row from the solution
        x = solution.get_values(0, num_columns-1)

        result = []
        #print for user
        for n in range (num_columns):
            if x[n] == 1:
                print "-----------------------------"
                print "%s = %10f" % (names[n], x[n])

                split = shlex.shlex(names[n], posix=True)
                split.whitespace += ','
                split.whitespace_split = True
                result.append(list(split))

        for c in range (previous_course,previous_course+course):
            wb2.get_sheet(0).write(0,c+3,course_list[c-previous_course])

        course_scheduled = []
        for i in range (len(result)):
            group_index     = int(result[i][1])
            room_index      = int(result[i][2])
            time_index      = int(result[i][3])
            course_index    = int(result[i][4])
            room_time_index = 10*Time*day*(room_index-1) + time_index

            capacity[room_index-1][time_index-1]                          -=
members_modified[group_index-1]

            wb2.get_sheet(1).write(time_index,previous_group+group_index+1,"*")

            if course_index not in course_scheduled:
                course_scheduled.append(course_index)

wb2.get_sheet(0).write(room_time_index,previous_course+course_index+2,"*")
        return True

    except CplexError:
        print ">>> NO SOLUTION FOUND <<<"
        return False

###############################################################################

# initializez the Cplex problem
def
populate(prob,course,group,group_registration,combination,combination_2_of_grou
p,members_modified,n_constraints,days):

    global names

    Input                                                                     =
problem(course,group,group_registration,combination,combination_2_of_group,memb
ers_modified,n_constraints,days)
    obj      = Input.obj
    lb       = Input.lb
    ub       = Input.ub
    types    = Input.types
    names    = Input.names
    fraction = Input.fraction
    senses   = Input.sense
    rhs      = Input.rhs
    # tell CPLEX to solve in minimization sense
    prob.objective.set_sense(prob.objective.sense.minimize)

    # add variable names and values to the problem
    prob.variables.add(obj = obj , lb = lb , ub = ub , types = types , names =
names)

    for n in range (len(fraction)):
        prob.linear_constraints.add(lin_expr = [[names,fraction[n]]] , senses =
[senses[n]] , rhs = [rhs[n]])

    print "^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^"
    print "NUMBER OF VARIABLES   :",len(obj)
    print ""
    print
"^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^"
    print "NUMBER OF CONSTRAINTS :",len(fraction)
    print ""

###############################################################################

def
problem(course,group,group_registration,combination,combination_2_of_group,memb
ers_modified,n_constraints,days):

    #fraction constraint
    fraction = [[0 for x in range (course*Time*days*room*group+1)] for y in
range  (Time*days*room  +  Time*days*group  +  (Time*days-Time+1)*group  +
group*course + course + n_constraints*Time*days*room + (Time*days - 4*Time
+1)*group + course*Time*day*room*group)]

#------------------------------------------------------------------------------

    #Constraint number 1, room capacity:
    for r in range (0, Time*days*room):
        for s_c in range (group):
            for c in range(course*r + course*Time*days*room*s_c , course*r +
course*Time*days*room*s_c + course):
                fraction[r][c] = members_modified[s_c]

#------------------------------------------------------------------------------
    # Constraint number 2, each group has one exam in one room in each
Time*day:
    for r in range (Time*days*room , Time*days*room + Time*days*group):
        s_r = int(math.floor((r-Time*days*room)/(Time*days)))
        for s_c in range (room):
            for c in range ((r - (Time*days*room + Time*days*s_r))*course +
course*Time*days*room*s_r + course*Time*days*s_c , (r - (Time*days*room +
Time*days*s_r))*course + course*Time*days*room*s_r + course*Time*days*s_c +
course):
                fraction[r][c] = 1

#------------------------------------------------------------------------------
    # Constraint number 3, 24 hr Time*day differences between each exams
    for r in range (Time*days*room + Time*days*group , Time*days*room +
Time*days*group + (Time*days-Time+1)*group):
        s_r      =      int(math.floor((r      -      (Time*days*room      +
Time*days*group))/(Time*days-Time+1)))
        for s_c in range (room):
            for c in range ((r - (Time*days*room + Time*days*group +
(Time*days-Time+1)*s_r))*course      +      course*Time*days*room*s_r      +
course*Time*days*s_c , (r - (Time*days*room + Time*days*group + (Time*days-
Time+1)*s_r))*course  +  course*Time*days*room*s_r  +  course*Time*days*s_c  +
course*Time):
                fraction[r][c] = 1

 #----------------------------------------------------------------------------
# Constraint number 4, for each group exactly 1 room-Time*day be allocated to
the course that he registered, and 0 room-Time*day if not registered
    for r in range (Time*days*room + Time*days*group + (Time*days-Time+1)*group
, Time*days*room + Time*days*group + (Time*days-Time+1)*group + course*group):
        s_r  =  int(math.floor((r  -  (Time*days*room  +  Time*days*group  +
(Time*days-Time+1)*group))/course))
        for s_c in range (Time*days*room):
            c  =  r  -  (Time*days*room  +  Time*days*group  +  (Time*days-
Time+1)*group) + (course*Time*days*room - course)*s_r + course*s_c
            fraction[r][c] = 1

#----------------------------------------------------------------------------
    # Constraint number 5, total room-Time*day-group corresponding each course
equal to total number of group registered in the course
    for r in range (Time*days*room + Time*days*group + (Time*days-Time+1)*group
+ course*group , Time*days*room + Time*days*group + (Time*days-Time+1)*group +
course*group + course):
        for s_c in range (Time*days*room*group):
            c  =  r  -  (Time*days*room  +  Time*days*group  +  (Time*days-
Time+1)*group + course*group) + course*s_c
            fraction[r][c] = 1

#----------------------------------------------------------------------------
    # Constraint number 6, to avoid splitting room-Time*day of each course
    s_r = -1
    i   = 0
    while i < course:
        if combination_2_of_group[i] > 0:
            for j in range (combination_2_of_group[i]):
                s_r += 1
                for r in range (Time*days*room + Time*days*group + (Time*days-
Time+1)*group + course*group + course + Time*days*room*s_r, Time*days*room +
Time*days*group  +  (Time*days-Time+1)*group  +  course*group  +  course  +
Time*days*room*s_r + Time*days*room):
                    c = (r - (Time*days*room + Time*days*group + (Time*days-
Time+1)*group  +  course*group  +  course  +  Time*days*room*s_r))*course  +  i  +
course*Time*days*room*combination[i][j][0]
                    fraction[r][c] = 1
                    c = (r - (Time*days*room + Time*days*group + (Time*days-
Time+1)*group  +  course*group  +  course  +  Time*days*room*s_r))*course  +  i  +
course*Time*days*room*combination[i][j][1]
                    fraction[r][c] = -1
            i += 1

        else:
            i +=1

#----------------------------------------------------------------------------
    # Constraint number 7, for each group exams are allocated in a way that no
more than 3 Time*day in a row
    for r in range (Time*days*room + Time*days*group + (Time*days-Time+1)*group
+  group*course  +  course  +  n_constraints*Time*days*room  ,  Time*days*room  +
Time*days*group  +  (Time*days-Time+1)*group  +  group*course  +  course  +
n_constraints*Time*days*room + (Time*days - 4*Time +1)*group):
        s_r  =  int(math.floor((r  -  (Time*days*room  +  Time*days*group  +
(Time*days-Time+1)*group      +      group*course      +      course      +
n_constraints*Time*days*room))/(Time*days - 4*Time +1)))
        for s_c in range (room):
            for  c  in  range  ((r  -  (Time*days*room  +  Time*days*group  +
(Time*days-Time+1)*group + group*course + course + n_constraints*Time*days*room
+  (Time*days  -  4*Time  +1)*s_r))*course  +  course*Time*days*room*s_r  +
course*Time*days*s_c , (r - (Time*days*room + Time*days*group + (Time*days-
Time+1)*group  +  group*course  +  course  +  n_constraints*Time*days*room  +
(Time*days   -   4*Time   +1)*s_r))*course   +   course*Time*days*room*s_r   +
course*Time*days*s_c + 4*course*Time):
                fraction[r][c] = 1
#-------------------------------------------------------------------------------
    # Constraint number 8, defining W
    for r in range (Time*day*room + Time*day*group + (Time*day-Time+1)*group +
group*course  +  course  +  n_constraints*Time*day*room  +  (Time*day  -  4*Time
+1)*group  ,  Time*day*room  +  Time*day*group  +  (Time*day-Time+1)*group  +
group*course  +  course  +  n_constraints*Time*day*room  +  (Time*day  -  4*Time
+1)*group + course*Time*day*room*group):
        fraction[r][r  -  (Time*day*room  +  Time*day*group  +  (Time*day-
Time+1)*group + group*course + course + n_constraints*Time*day*room + (Time*day
- 4*Time +1)*group)] +1
        #math.floor((r  -  Time*day*room  +  Time*day*group  +  (Time*day-
Time+1)*group + group*course + course + n_constraints*Time*day*room + (Time*day
- 4*Time +1)*group)/course) + 1
        fraction[r][course*Time*day*room*group] = -1

#-------------------------------------------------------------------------------
                # Write to excel file
                #for i in range (len(fraction)):
                #    for j in range(size(fraction)/len(fraction)):
                #        wb2.get_sheet(0).write(i,j,fraction[i][j])

                #wb2.save('C:\\Users\\6R16R\\Desktop\\fraction.xls')

#-------------------------------------------------------------------------------
    objective   =  ["" for x in range ((course*Time*day*room*group)+1)]
    lower_bound =  ["" for x in range ((course*Time*day*room*group)+1)]
    upper_bound =  ["" for x in range ((course*Time*day*room*group)+1)]
    var_types   = ["I" for x in range ((course*Time*day*room*group)+1)]
    var_names   =  ["" for x in range ((course*Time*day*room*group)+1)]

    for s in range (group):
        for r in range (room):
            for t in range (Time*day):
                for c in range (course):
                    objective[course*Time*day*room*s   +   course*Time*day*r   +
course*t + c] = cost[r][t%Time]
                    lower_bound[course*Time*day*room*s   +   course*Time*day*r   +
course*t + c] = 0
                    upper_bound[course*Time*day*room*s   +   course*Time*day*r   +
course*t + c] = 1
                    var_names[course*Time*day*room*s   +   course*Time*day*r   +
course*t + c]  = "X,"+str(s+1)+","+str(r+1)+","+str(t+1)+","+str(c+1)

    objective[(course*Time*day*room*group)] = 1000
    lower_bound[(course*Time*day*room*group)] = 0
    upper_bound[(course*Time*day*room*group)] = 30*3
    var_names[(course*Time*day*room*group)] = "W"
#-------------------------------------------------------------------------------
    rhs    = ["" for x in range (Time*day*room + Time*day*group + (Time*day-
Time+1)*group + group*course + course + n_constraints*Time*day*room + (Time*day
- Time*(3+1) +1)*group + course*Time*day*room*group)]
    print "rhs", len(rhs)
    sense = "L" * (Time*day*room + Time*day*group + (Time*day-Time+1)*group) +
"E" * (group*course + course + n_constraints*Time*day*room) + "L" * ((Time*day
- Time*(3+1) +1)*group + course*Time*day*room*group)
```

```
    print "sense", len(sense)

    for r in range (room):
        for t in range (Time*days):
            rhs[Time*days*r + t] = capacity[r][t]

    for  i  in  range  (Time*days*room  ,  Time*days*room  +  Time*days*group  +
((Time*days)-Time+1)*group):
        rhs[i] = 1.0

    for c in range (course):
        total = 0
        for s in range (group):
            rhs[(Time*days*room + Time*days*group + ((Time*days)-Time+1)*group
+ course*s )+ c] = group_registration[c][s]
            if group_registration[c][s] != "":
                total += group_registration[c][s]
        rhs[(Time*days*room  +  Time*days*group  +  ((Time*days)-Time+1)*group  +
group*course) + c] = total

    for   i   in   range   (Time*days*room   +   Time*days*group   +   ((Time*days)-
Time+1)*group  +  group*course  +  course  ,  Time*days*room  +  Time*days*group  +
((Time*days)-Time+1)*group      +      group*course      +      course      +
n_constraints*Time*days*room):
        rhs[i] = 0.0

    for   i   in   range   (Time*days*room   +   Time*days*group   +   ((Time*days)-
Time+1)*group   +   group*course   +   course   +   n_constraints*Time*days*room   ,
Time*days*room + Time*days*group + ((Time*days)-Time+1)*group + group*course +
course + n_constraints*Time*days*room + ((Time*days) - Time*(3+1) +1)*group):
        rhs[i] = 3.0

    for i in range (Time*day*room + Time*day*group + ((Time*day)-Time+1)*group
+  group*course  +  course  +  n_constraints*Time*day*room  +  ((Time*day)  -
Time*(3+1)  +1)*group  ,  Time*day*room  +  Time*day*group  +  ((Time*day)-
Time+1)*group  +  group*course  +  course  +  n_constraints*Time*day*room  +
((Time*day) - Time*(3+1) +1)*group + course*Time*day*room*group):
        rhs[i] = 0.0

    return
Problem(objective,var_names,var_types,lower_bound,upper_bound,sense,rhs,fractio
n)

###############################################################################

one_by_one_solver()
```

Appendix II. Real Dataset Output for Model Two (Grouping Approach)

Table 34. Student-Course Registrations

Student\Course	ANAT1010	BIOL1010	CHEM1011	CLAS1100X	COMM1010	CSCI1100	ENGI1101	ENGI2200	ERTH1060	HIST1501	HIST1701	MATH1000	MUSC1020	POLI1020	STAT1060	THEA1000X	IENG3316	IENG4529	MECH4010	MECH4660	MECH4851
1	1		1			1						1									
2		1		1	1		1		1		1										
3								1		1			1	1	1	1					
4																	1	1	1	1	1
5	1												1								
6		1		1	1		1		1		1										
7																	1	1	1	1	1
8		1		1	1		1		1		1										
9								1		1			1	1	1	1					
10																	1	1	1	1	1
11		1		1	1		1		1		1										
12																	1	1	1	1	1
13		1		1	1		1		1		1										
14								1		1			1	1	1	1					
15	1												1								
16		1	1								1	1					1				1
17		1		1	1		1		1		1										
18		1	1								1	1					1				1
19		1	1								1	1					1				1
20		1		1	1		1		1		1										
21								1		1			1	1	1	1					
22		1	1								1	1					1				1
23	1			1		1		1											1		
24	1			1		1		1											1		
25		1			1					1				1	1						1
26		1			1					1				1	1						1
27		1		1	1		1		1		1										
28								1		1			1	1	1	1					
29		1	1								1	1					1				1
30		1		1	1		1		1		1										
31								1		1			1	1	1	1					
32		1	1								1	1					1				1
33	1			1		1		1											1		
34		1			1					1				1	1						1
35		1			1					1				1	1						1
36		1		1	1		1		1		1										
37								1		1			1	1	1	1					
38		1	1								1	1					1				1
39	1			1		1		1											1		
40		1			1					1				1	1						1
41		1		1	1		1		1		1										
42								1		1			1	1	1	1					
43		1	1								1	1					1				1
44	1			1		1		1											1		
45		1		1	1		1		1		1										
46								1		1			1	1	1	1					
47		1	1								1	1					1				1
48	1																		1		
49		1		1	1		1		1		1										
50	1																		1		
51								1		1			1	1	1	1					
52	1			1		1		1											1		
53	1			1		1		1											1		
54		1		1	1		1		1		1										
55								1		1			1	1	1	1					
56			1						1							1			1		1
57	1	1																			
58	1			1		1		1											1		
59			1						1							1			1		1
60		1		1	1		1		1		1										
61	1			1		1		1											1		
62			1						1							1			1		1
63				1	1	1	1														
64				1	1	1	1														
65	1		1			1						1									
66	1			1		1		1											1		
67			1						1							1			1		1
68				1	1	1	1														
69		1		1	1		1		1		1										
70	1			1		1		1											1		
71			1						1							1			1		1
72			1						1							1			1		1
73		1											1								
74			1						1							1			1		1
75	1						1				1	1									
76			1						1							1			1		1
77			1						1							1			1		1
78		1		1	1		1		1		1										
79	1			1		1		1											1		
80			1						1							1			1		1
81		1			1																
82	1		1																		
83			1						1							1			1		1
84	1													1							
85										1	1										
86			1						1							1			1		1
87		1		1	1		1		1		1										
88			1						1							1			1		1
89			1				1						1					1			
90					1					1					1						
91							1	1				1									
92			1						1							1			1		1
93		1		1	1		1		1		1										
94			1						1							1			1		1
95	1																			1	
96											1		1					1			
97			1						1							1			1		1
98							1	1	1												
99			1						1							1			1		1
100		1		1	1		1		1		1										

Table 35. Members of the Groups using the Second Model for Real Dataset

Group	Group 1	Group 2
Number of Members	2	20
Member	0-63	1-3-11-15-21-26-35-45-48-52-56-71-77-81-83-86-87-92-97-99
Group	Group 3	Group 4
Number of Members	11	4
Member	2-7-19-23-36-44-49-66-79-88-91	4-62-78-94
Group	Group 5	Group 6
Number of Members	2	9
Member	5-70	6-30-43-54-61-67-80-93-98
Group	Group 7	Group 8
Number of Members	2	12
Member	8-29	9-16-34-37-42-50-58-65-69-76-85-95
Group	Group 9	Group 10
Number of Members	4	1
Member	10-28-38-53	12
Group	Group 11	Group 12
Number of Members	5	17
Member	13-22-46-60-74	14-18-27-33-39-51-55-59-64-68-72-75-82-84-89-90-96
Group	Group 13	Group 14
Number of Members	1	3
Member	17	20-47-73
Group	Group 15	Group 16
Number of Members	1	1
Member	24	25
Group	Group 17	Group 18
Number of Members	1	1
Member	31	32
Group	Group 19	Group 20
Number of Members	1	1
Member	40	41
Group	Group 21	
Number of Members	1	
Member	57	

Table 36. Course Selection for Each Group for Real Dataset

	ANAT1010	BIOL1010	CHEM1011	CLAS1100X	COMM1010	CSCI1100	ENGI1101	ENGI2200	ERTH1060	HIST1501	HIST1701	MATH1000	MUSC1020	POLI1020	STAT1060	THEA1000X	IENG3316	IENG4529	MECH4010	MECH4660	MECH4851
Group 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Group 2	1	0	1	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
Group 3	1	0	0	0	0	0	0	1	0	0	0	0	1	1	1	0	0	0	0	0	0
Group 4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Group 5	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Group 6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
Group 7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
Group 8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
Group 9	0	1	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Group 10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Group 11	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
Group 12	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
Group 13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
Group 14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
Group 15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
Group 16	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
Group 17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Group 18	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
Group 19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Group 20	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
Group 21	1	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0

Table 37. Examinations Schedule using the Second Model for Real Dataset

			ANAT1010	BIOL1010	CHEM1011	CLAS1100X	COMM1010	CSCI1100	ENGI1101	ENGI2200	ERTH1060	HIST1501	HIST1701	MATH1000	MUSC1020	POLI1020	STAT1060	THEA1000X	IENG3316	IENG4529	MECH4010	MECH4660	MECH4851
Gym	M	8:30:00-11:30					*		*	*													
		12:00:00 -3:00																					
		3:30-6:30																					
	T	8:30:00-11:30	*	*				*											*		*		
		12:00:00 -3:00																					
		3:30-6:30																					
	W	8:30:00-11:30			*	*							*		*							*	
		12:00:00 -3:00																					
		3:30-6:30																					
	R	8:30:00-11:30																					
		12:00:00 -3:00																					
		3:30-6:30																					
	F	8:30:00-11:30									*	*				*		*		*			*
		12:00:00 -3:00																					
		3:30-6:30																					
	S	8:30:00-11:30												*			*						

Appendix III. Real Dataset Output for Model Five

Table 38. Members of the Groups of the First Department using the Fourth Model for Real Dataset

Groups (First Dep.)	Group 1	Group 2
Number of Members	20	13
Member	0-2-6-9-12-16-21-27-29-32-35-43-47-49-51-53-54-59-61-62	1-4-11-14-22-26-30-40-48-55-58-25-36
Groups (First Dep.)	Group 3	Group 4
Number of Members	6	7
Member	5-18-23-33-3-42	7-13-28-38-45-15-19

Table 39. Members of the Groups of the Second Department using the Fourth Model for Real Dataset

Group(Second Dep.)	Group 1	Group 2
Number of Members	7	17
Member	1-23-30-34-0-24-15	2-11-17-20-22-26-31-33-36-3-10-5-7-8-18-28-9

Table 40. Course Selection for Each Group of the First Department for Real Dataset

1	ANAT1010	BIOL1010	CHEM1011	CLAS1100X	COMM1010	CSCI1100	ENGI1101	ENGI2200	ERTH1060	HIST1501	HIST1701	MATH1000	MUSC1020	POLI1020	STAT1060	THEA1000X
Group 1	1	0	1	0	0	0	1	0	0	1	0	1	0	0	0	0
Group 2	1	0	0	0	0	0	0	1	0	0	0	0	1	1	1	0
Group 3	0	1	0	1	0	0	1	0	1	0	0	0	0	0	0	0
Group 4	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	1

Table 41. Course Selection for Each Group of the Second Department for Real Dataset

2	IENG3316	IENG4529	IENG4579	MECH4010	MECH4660	MECH4851
Group 1	1	1	1	0	0	0
Group 2	0	0	0	1	1	1

Table 42. Examinations Schedule using the Fifth Model for Real Dataset

			ANAT1010	BIOL1010	CHEM1011	CLAS1100X	COMM1010	CSCI1100	ENGI1101	ENGI2200	ERTH1060	HIST1501	HIST1701	MATH1000	MUSC1020	POLI1020	STAT1060	THEA1000X	IENG3316	IENG4529	IENG4579	MECH4010	MECH4660	MECH4851
Gym	M	8:30:00-11:30		*	*		*			*									*			*		
		12:00:00 -3:00																						
		3:30 - 6:30																						
	T	8:30:00-11:30				*		*				*			*					*			*	
		12:00:00 -3:00																						
		3:30 - 6:30																						
	W	8:30:00-11:30									*		*	*		*					*			*
		12:00:00 -3:00																						
		3:30 - 6:30																						
	R	8:30:00-11:30																						
		12:00:00 -3:00																						
		3:30 - 6:30																						
	F	8:30:00-11:30	*															*						
		12:00:00 -3:00																						
		3:30 - 6:30																						
	S	8:30:00-11:30							*								*							