Faculty of Computer Science and Information Technology

Computerised Multi-Objective Faculty Exam Timetabling Problem

Phang Min Hui

Master of Science
2019

Computerised Multi-Objective Faculty Exam Timetabling Problem

Phang Min Hui

A thesis submitted

In fulfilment of the requirements for the degree of Master of Science

(Computer Science)

Faculty of Computer Science and Information Technology
UNIVERSITI MALAYSIA SARAWAK
2019

DECLARATION

I, Phang Min Hui from Faculty of Computer Science and Information Technology, UNIMAS hereby declare that the work entitled ‘Computerised Multi-Objective Faculty Exam Timetabling Problem’ is the original work of the author alone. Except where due reference or acknowledgement is made explicitly in the text, no part of the content of this work is copied from any other sources. The thesis has not been accepted for any degree and is not concurrently submitted in candidature for any other degree.

……..……

Signature

Name of Student:	Phang Min Hui
Matric No:	16020134
Faculty:	Faculty of Computer Science and Information Technology, Sarawak
Date:	

ACKNOWLEDGEMENT

First of all, I would like to express my great appreciation to my supervisor, Dr. Sze San Nah for her valuable and constructive advice, guidance and encouragement throughout this research project. Besides that, I am very grateful to her for her willingness to share her knowledge in this research project. Moreover, special thanks to my great co-supervisor, Dr. Chiew Kang Leng for his patience and valuable time in implicitly sharing his knowledge and giving advice in my hard times throughout this research.

I also wish to extend my appreciation to Universiti Malaysia Sarawak for the Zamalah Vice Chancellor Research Award which provided me with a full scholarship and made my Master's degree a reality. Meanwhile, thanks to the Faculty of Computer Science and Information Technology for providing a learning environment that allowed me to focus on my research until its completion. I would also like to thank the faculty management staff for their important role in their contribution, time and effort in providing me all the data needed for this research.

In addition, thanks to all my friends and research mates for giving their valuable advice, ideas and support throughout this research project. Last but not least, thanks to my family for their encouragement and the power they have given to me.

ABSTRACT

This research focuses on the final examination timetabling at the Faculty of Computer Science and Information Technology (FCSIT), Universiti Malaysia Sarawak (UNIMAS). In UNIMAS, each faculty needs to schedule a final examination timetable every semester. The large number of students and limited resources in each faculty may increase the problem complexity when arranging the final examination timetable. In this study, there is one existing system (FESS 1.0) to generate a clash-free timetable. However, student sectioning, room utilisation, continuous examination gaps and priority courses were not considered. The main objective of this research is to design and build a computationally bounded multi-objective two-stage heuristic algorithm to optimise examination room utilisation. The first stage, course grouping, is mainly to minimise the problem size and clash-free constraints. All the courses are divided into a smaller number of course groups. The course groups are then eased into the second stage of timeslot-room allocation. During the allocation, each course is allocated at 'best fit room' in order to maximise the room utilisation and minimise the student sectioning. A few real datasets were collected and experimented with the proposed solution. Overall, the proposed solution is proven to outperform the existing solution in terms of room utilisation, student sectioning, continuous examination and priority course. Besides that, it is also able to accommodate the priority courses constraints well in order to obtain earlier examination dates. Subsequently, a sensitivity analysis was conducted. The increment and decrement in the course size, course-student enrolment size and room size were tested respectively in the solution. All the sensitivity results proved that the proposed solution is effective and robust to solve different types of datasets.

Keywords: Examination timetabling, two-stage heuristic, student sectioning, room utilisation

Penjadualan Berkomputer Pelbagai Objektif Masalah Jadual Waktu Peperiksaan Fakulti

ABSTRAK

Kajian ini memberi tumpuan kepada penjadualan peperiksaan akhir di Fakulti Sains Komputer dan Teknologi Maklumat (FSKTM), Universiti Malaysia Sarawak (UNIMAS). Di UNIMAS, setiap fakulti perlu menjana jadual peperiksaan akhir setiap semester. Bilangan pelajar yang besar dan sumber yang terhad di setiap fakulti meningkatkan kerumitan masalah semasa penjadualan peperiksaan akhir. Dalam kajian kes ini, terdapat satu sistem yang sedia ada (FESS 1.0) untuk menjana jadual tanpa masalah pertembungan. Walau bagaimanapun, pembahagian pelajar, penggunaan bilik dan jurang peperiksaan berterusan tidak dipertimbangkan. Objektif utama kajian ini adalah untuk merekabentuk dan membina satu algoritma heuristik dua-peringkat yang komputasi terikat dan berbilang objektif untuk mengoptimumkan penggunaan bilik peperiksaan. Di peringkat pertama, iaitu pengumpulan kursus, bertujuan mengurangkan saiz masalah dan kekangan tanpa pertembungan. Semua kursus akan dibahagikan ke dalam kumpulan kursus yang lebih kecil. Kemudian, kumpulan kursus akan digunakan di peringkat kedua iaitu peruntukan bilik-slot masa. Semasa peruntukan, setiap kursus akan diperuntukkan di 'bilik tersesuai' untuk memaksimumkan penggunaan bilik dan mengurangkan pembahagian pelajar. Beberapa dataset yang sebenar telah dikumpul dan diuji dengan penyelesaian yang dicadangkan. Secara keseluruhannya, penyelesaian yang dicadangkan telah mengatasi penyelesaian yang sedia ada dari segi penggunaan bilik, pembahagian pelajar, jurang peperiksaan berterusan dan sebagainya. Di samping itu, penyelesaian yang dicadangkan itu juga dapat menampung kekangan kursus keutamaan dengan baik untuk mendapatkan tarikh peperiksaan yang lebih awal. Selain itu, analisis kepekaan juga dijalankan. Peningkatan dan penurunan antara saiz bilik, saiz

enrolmen pelajar dan saiz bilik masing-masing telah diuji dalam penyelesaian. Semua hasil kepekaan membuktikan bahawa penyelesaian yang dicadangkan adalah berkesan dan mantap untuk menyelesaikan pelbagai jenis dataset.

Kata kunci: *Penjadualan peperiksaan, kaedah heuristik, pembahagian pelajar, penggunaan bilik*

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

1.1 Introduction

Examination timetabling is assigning a set of examinations into a set of timeslots and rooms with the aim of satisfying a set of constraints (Mandal & Kahar, 2015). It is an NP-hard problem which requires a large amount of computation to solve the scheduling problem (Parker & Rardin, 1988). The constraints vary from institution to institution. There are two types of constraints i.e. hard constraints and soft constraints. Hard constraints must be satisfied in order to get a feasible solution. Meanwhile, fulfilling soft constraints is optional but usually the quality of the solutions are identified based on the soft constraints.

The educational timetabling problem started in the 1960s. The timetabling problem can be solved manually when the problem size is small. Nowadays, the student population is increasing from semester to semester. The increasing student population causes the problem size to become bigger. Not only that, the examination timetabling problem becomes more complex when there are limited timeslots and venue resources available. The different factors which are related to each other may increase the complexity of the examination timetabling problem. Thus, the study of examination timetabling is important and critical to solve the problem.

Over the last 50 years, various methods have been applied to solve the examination timetabling problem. These methods have been applied into several timetabling software. However, many institutions still solve their timetabling problems manually. This is because different institutions have different constraints. Although most of the institutions have the

same general constraints, unique constraints in each institution result in differing sets of problems for each institution. There is no software that can fit all the constraints for every institution. Thus, most of the institutions prefer to solve the examination timetabling problem manually. When the problem size is big, the manual effort of solving the problem could be very time-consuming and labour-intensive.

1.2 Problem Statement

In this research, the faculty final examination timetabling problem at the Faculty of Computer Science and Information Technology (FCSIT), Universiti Malaysia Sarawak (UNIMAS) has been studied.

Similar to other institutions, a final examination timetable in UNIMAS will be scheduled for every semester. Examinations for generic and elective courses will be scheduled by the UNIMAS management. However, the examination timetable for each faculty course will be set by the faculty themselves. Therefore, the FCSIT administrator needs to schedule the final examination timetable for every semester.

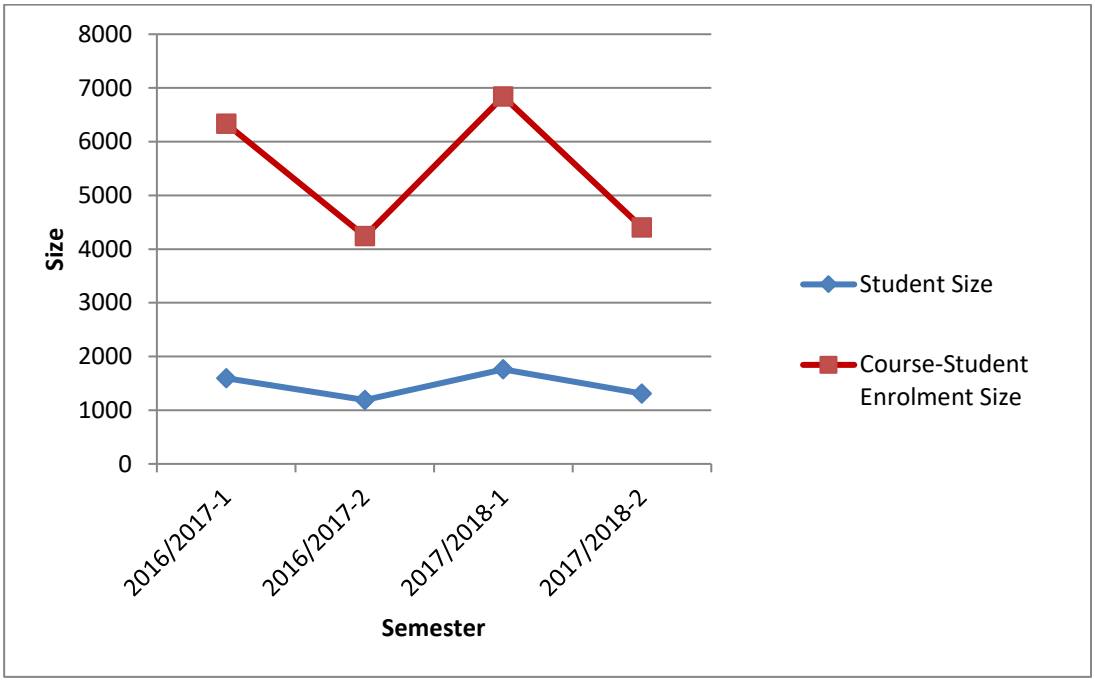


Figure 1.1: Statistic on course-student enrolment at FCSIT

As shown in Figure 1.1, the student size at FCSIT has been increasing from semester to semester. When the student number increases, the size of course-student enrolment in the faculty also increases. However, the timeslots and examination rooms for the final examination still remain the same. The scheduling process becomes harder with the limited resources and large course-student enrolment size. The clash problem faced by students also increases as the course-student enrolment in different courses rises. If the final examination timetable is scheduled manually, it would be very time consuming to generate a clash-free timetable.

Previously, FCSIT used the Final Examination Scheduling System (FESS 1.0) to solve the final examination timetabling problem. The system uses a heuristic method that focused on the feasibility of the timetable. Students would not experience any clashes in their timetable during the final examination. However, the final examination timetable generated by FESS 1.0 does not focus on room utilisation, where most of the courses are split into

many exam venues. The unutilised room capacity is also very high. A good timetable is not only dependent on the feasibility of the solution, the quality of the solution is also very important. The solution quality has to take into account factors such as room utilisation, student sectioning, continuous examinations and the consideration of *priority course*.

1.3 Research Objectives

The objective of this research are:

(a) Develop a computationally bounded multi-objective two-stage heuristic algorithm for the final examination timetabling problem.

(b) Develop a simulation model of the faculty examination problem.

(c) Conduct computational and sensitivity analyses on the proposed algorithm on:

- Room utilisation
- Student sectioning
- Continuous examination
- Priority course

1.4 Research Scope

The research scope focuses on the items below:

(a) Focus on the data from FCSIT. Data from other faculties in UNIMAS are also collected, such as the Faculty of Social Sciences (FSS), Faculty of Economics and Business (FEB), Faculty of Engineering (FENG) and Faculty of Cognitive Sciences and Human Development (FCSHD).

(b) The simulation model in this study focuses on the final examination timetable for faculty core courses only, as the faculty elective courses and university generic courses are pre-scheduled by the university management.

(c) Computational and sensitivity analysis: Besides the comparison study, the proposed algorithm is also evaluated in terms of room utilisation, student sectioning for each course, continuous examinations and the distribution of priority courses during the examination period.

1.5 Research Significance

The significance of this research is to develop an applicable two-stage heuristic algorithm for the final examination timetabling problem. This clash-free solution will replace the current scheduling system. This new hybrid heuristic algorithm improves the quality of the timetable in different aspects such as room utilisation, continuous examination and priority courses. Since continuous examination is considered in this research, students will have a better timetable solution throughout the final examination period. Moreover, this algorithm has been successfully implemented at FCSIT, UNIMAS since Semester 1, 2016/2017.

1.6 Thesis Outline

The following chapters are outlined as follows:

Chapter 2: Provides a review of examination timetabling historical development. Problems and solving methods are discussed in different categories.

Chapter 3: Explains the case study of the faculty final examination timetabling problem in detail. The modelling and formulation of the problem are described in this chapter.

Chapter 4: Describes the proposed two-stage heuristic method in detail. This chapter also contains an explanation of the different factors which will improve the solution quality.

Chapter 5: Describes the dataset specifications. The computational analysis of different datasets is conducted by comparing timetable solutions solved by the two-stage heuristic method with the existing method. The result of the sensitivity analysis is also discussed in this chapter.

Chapter 6: Concludes the findings in this research. The achievements of this research are presented. Lastly, ideas for some future plans that may improve the solution are also discussed in this chapter.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter is going to review the examination timetabling problem. The exact solution, heuristic, metaheuristic and hyper-heuristic methods have been studied in solving the examination timetabling problem.

2.2 Overview of Educational Timetabling

Educational timetabling problems include university examination timetabling, university course timetabling and school timetabling. These problems are combinatorial optimisation problems which require the allocation of resources to satisfy a set of constraints (Pillay, 2016). Generally, the venue resources used in schools are fixed compared with universities. In universities, the venue resources used during lecture hours are greater than during the examination period. Therefore, the allocation of resources in a university examination timetabling is more complicated than others.

2.3 Examination Timetabling

The university examination timetabling algorithm is typically a hard problem in allocating exams into a limited number of periods and resources (Carter et al., 1996; Burke & Newall, 2004; Asmuni, Burke, Garibaldi & McCollumn, 2004; Eley, 2006). Most

institutions will need to solve this problem twice a year. Therefore, it is a critical issue faced by most academic institutions.

The biggest problem of examination timetabling is fitting all the exams into a limited number of rooms (Carter et al., 1996). The capacity of rooms for exams is usually half of the original capacity in course timetabling. The limited number of rooms and capacity greatly increase the complexity of the timetabling problem.

2.4 Existing Timetabling Methods

The timetabling problem is an NP-hard problem which requires a large amount of computation to solve the problem (Abdul-Rahman et al., 2014). As shown in Figure 2.1, over the last 50 years, numerous methods have been proposed to tackle the examination timetabling problem.

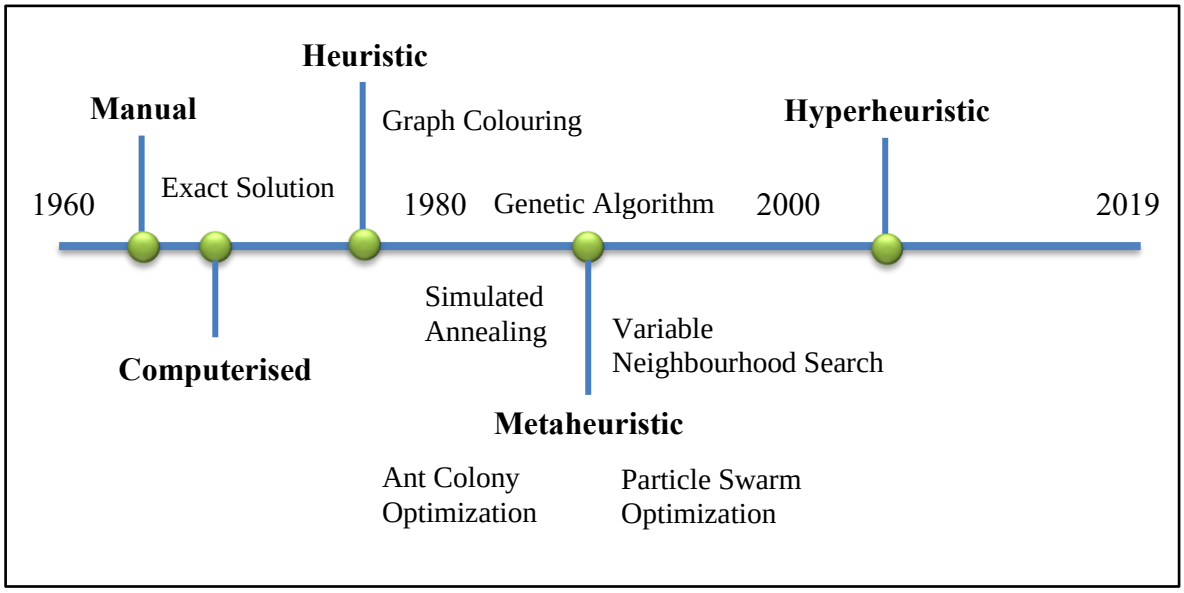


Figure 2.1: Chronological progress of solving the examination timetabling problem

2.4.1 Exact Solution

The examination timetabling problem is in assigning a set of exams into a set of timeslots. Generally, the objective in solving the examination timetabling problem is to satisfy all hard constraints and minimising the violations of soft constraints while generating a feasible solution. Therefore, an exact solution is introduced to solve the examination timetabling problem. There are several exact solutions that are applied to solve the examination timetabling problem, such as constraint logic programming (Boizumault et al., 1996), integer programming (Cataldo et al., 2017), integer linear programming (Aizam & Uvaraja, 2015; Jamaluddin & Aizam, 2016), linear mixed-integer programming (Bergamann et al., 2014), fuzzy integer linear programming (Chaudhari & De, 2013), and column generation (Wijgers & Hoogeveen, 2006). Most of these exact solutions are curriculum-based. Therefore, the exact solution is suitable only for small problem sizes.

2.4.2 Heuristic

In its early years, the examination timetabling problem was solved manually. The examination timetabling problem became more complicated as the problem size got bigger. Therefore, a heuristic was introduced to produce an optimal solution by satisfying all the hard constraints of the timetable. The graph colouring method is one of the most common heuristic methods applied in examination timetabling (Burke & Newall, 2004; Pillay & Ozcan, 2017). It is an approach where the difficult examinations will be scheduled first and only then will the easier examinations be scheduled. A backtracking procedure is needed to set a particular exam into the timetable. However, an unscheduled examination might happen due to a clash with the scheduled examination.

2.4.3 Metaheuristic

Nowadays, most examination timetabling problems are large and very complex. Some of the early methods are computationally expensive to solve these problems. Most of these problems are classified as combinatorial optimisation problems. Therefore, metaheuristic is introduced to solve the combinatorial optimisation problems. Metaheuristic is a high-level independent algorithm that provides a set of guidelines to develop heuristic optimisation algorithms (Sörensen et al., 2017). It has been studied by many researchers and is very successful in solving examination timetabling problems. Examples of metaheuristic techniques are Simulated Annealing (Chmait & Challita, 2013), Genetic Algorithm (Pillay & Banzhaf, 2010; Hosny & Al-Olayan, 2014), Ant Colony Optimisation (Dowsland & Thompson, 2005; Eley, 2006; Chmait & Challita, 2013), Particle Swarm Optimisation (Marie-Sainte, 2017) and Variable Neighbourhood Search (Burke et al., 2010). Even though metaheuristic can generate better quality solutions, it requires more computational expense than simple heuristic.

2.4.4 Hyper-heuristic

Hyper-heuristic, a higher level of abstraction than the metaheuristic method was introduced to solve the examination timetabling problem (Burke et al., 2003). By making minor adjustments, it can solve more than one problem instances with different constraints. The real automated examination timetabling technique with hyper-heuristic is applied in society. However, this is not a practical solution as a new system has to be developed when there are any different requirements in the examination timetabling problem. Besides that, the possibility of high runtimes has to be identified and distributed with hyper-heuristics

(Pillay, 2016). This may require expensive equipment to implement and distribute hyper-heuristics.

2.5 Practical Cases

There are currently many research which have been conducted on the examination timetabling problem. However, a successful research is one where it can be tested and validated with a real timetabling problem, which can produce impressive results on the tested data.

There are a few successful research of practical methods with real examination timetabling problems. Kendall and Hussin (2004) used a tabu search based hyper-heuristic to solve the examination timetabling problem at MARA University of Technology (UiTM). UiTM has a large number of students which increases the complexity of the timetabling problem. However, this research was able to produce a feasible examination timetable without any modification to the hyper-heuristic framework.

Ayob et al. (2007) successfully implemented their research method on solving the practical examination timetabling problem at Universiti Kebangsaan Malaysia (UKM). The Greedy-Least Saturation Degree (G-LSD) heuristic was applied in the research to solve the real examination timetabling problem. The research successfully solved the timetabling problem with the objective of minimising the number of students having consecutive exams on the same day and on the next day.

Kahar and Kendall (2010) successfully implemented a constructive heuristic to solve the final examination timetabling problem at Universiti Malaysia Pahang (UMP). This

research successfully achieved the constraints of minimising the distance between examination rooms and the splitting of exams across several rooms.

Marie-Sainte (2017) used the Particle Swarm Optimisation (PSO) technique to solve the university examination timetabling problem at the College of Computer and Information Sciences at King Saud University. The timetabling problem faced by this university was the clashing issue where students may have more than one exam at a time. By using the PSO technique, the problem was fixed successfully.

2.6 Conclusion

Based on the review of previous research, examination timetabling is assigning a set of examinations into a set of timeslots with various types of constraints. Most of the timetabling problems are solved by a heuristic approach. The practical cases show that different institutions have different constraints. Each practical case mostly only focused on one objective function of the timetabling problem, such as to maximise room utilisation and minimise continuous examinations. Therefore, different techniques are applied to solve different constraints.

In our case, the study of the examination timetabling problem is to focus on more than one constraint. The most common and important constraint is for students to have one examination at a time. Since there are a lot of constraints, a two-stage heuristic method is proposed to solve the examination timetabling problem. The two-stage heuristic can reduce the complexity of the problem and easily accommodate abundant constraints. In this research, a *student-based* solution will be applied to generate a feasible solution. The *curriculum-based* solution is not suitable to solve the examination timetabling problem as it cannot

overcome the repeater problem which may lead to the clashing issue. This research will also focus on other constraints such as students' continuous exams, room utilisation, and splitting exams across several rooms. These constraints will be justified with the quality of the examination timetable.

CHAPTER 3

MODELLING AND FORMULATION

3.1 Introduction

This chapter will discuss the case study of this research. The real problem of the final examination timetabling at the Faculty of Computer Science and Information Technology (FCSIT), Universiti Malaysia Sarawak (UNIMAS), is being used for this study. The modelling and formulation will then be conducted accordingly.

3.2 Problem Description

There are five undergraduate programmes offered at FCSIT. They are Computational Science (CS), Information Systems (IS), Multimedia Computing (MC), Network Computing (NC), and Software Engineering (SE). All five programmes are four-year programmes with a total of seven semesters on campus and one semester of industrial training. In each semester, FCSIT will offer around 50 courses for students. All the courses offered are different from semester to semester. Thus, in every semester, the final examination timetable needs to be scheduled for all the different programmes. Previously, the final examination timetable was scheduled manually by the scheduling planner, which may take around two weeks. Starting from Semester 2 (2014/2015), the faculty utilised a final examination scheduling system to assist in the timetabling process. The scheduling system improved the timetabling process. A final examination timetable can be produced in around one minute with no clash issues. However, there is a problem of an empty timeslot within the examination period. It is because

the course group in other timeslots do not fit into the empty timeslot. In order to overcome the empty timeslot, the administrator needs to manually edit the generated final examination timetable. Therefore, the empty timeslot is not a practical solution which can be prevented, and the examination period can be shortened. Furthermore, there are several issues with the current timetabling process:

(a) Nonconsideration of priority courses

Some courses' final exam ought to be conducted earlier due to the following reasons: (i) Students are going for industrial training right after their exam; and (ii) Lecturers are away for training or long leave (such as study leave or maternity leave). As the priority course is take into consideration, students can prepare for their industry training as early as possible. Similarly, lecturers can mark the papers as the examination can finish earlier. If the priority course is not considered during the scheduling process, the scheduler planner may need to manually adjust the timetable to give priority to the related courses.

(b) Nonconsideration of the number of student sectioning

Student sectioning is a process when the room cannot accommodate all the students of a course. If the sectioning process of a course is higher, this means that the course will need more rooms for the examination. Previously, the timetable consisted of some courses with a high number of student sectioning (as shown in Figure 3.1). The administrator was required to make more amendments in order to minimise the student sectioning. If the amendment was not included, this would increase the number of invigilators on duty. Therefore, the high number of student sectioning must be prevented.

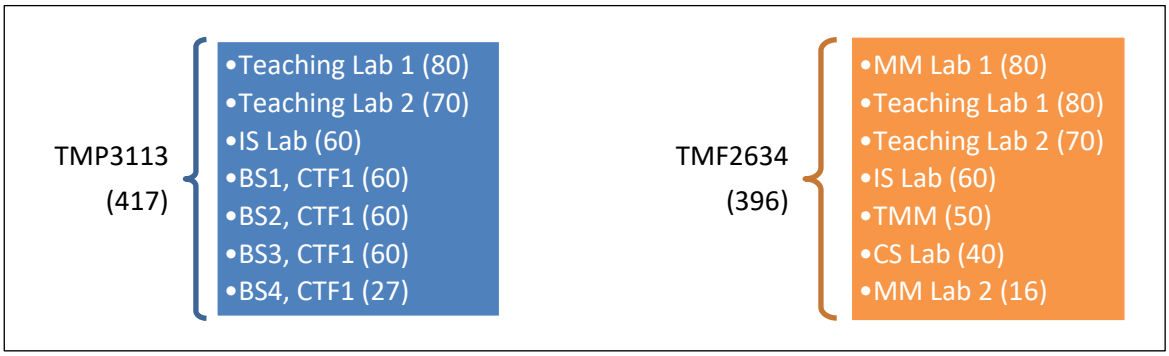


Figure 3.1: Student sectioning of course

(c) Nonconsideration of continuous examination

Continuous examination means that the student will sit for their examination from one session to another session continuously. As shown in Figure 3.2, there are nine occurrences of three continuous examinations and 18 occurrences of two continuous examinations. These are not practical solutions because students do not have enough time to revise for the next examination. If this situation is more serious than expected, students will need to check the timetable and give feedback to the administrator. The administrator will need to change the timetable as requested by the students. Therefore, continuous examinations for students must be prevented.

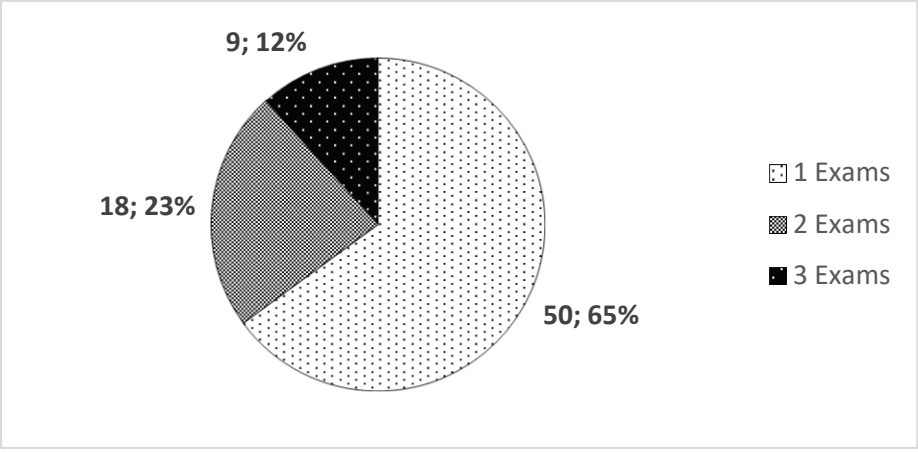


Figure 3.2: Number of occurrences of continuous examination

3.2.1 Venue Resources

There are two types of venue resources in UNIMAS i.e. the faculty's own rooms and shared rooms. The shared rooms are shared among all the faculties in UNIMAS. They are managed at the university level. Therefore, they are a limited resource where the faculties only can use them for selected timeslots. Meanwhile, the rooms owned by the faculty are available all the time, exclusively for their own use. Table 3.1 shows the details of venue resources for FCSIT.

Table 3.1: Venue resources for FCSIT

Management Team	Venue Usage Authority	Availability	Quantity	Capacity
University Level	Shared among faculties	Limited	4	≥ 300
			2	200 to 250
			6	≤ 100
Faculty Level	Owned by FCSIT	All the time	1	100
			6	60 to 80
			1	30

3.2.2 Timeslots

Generally, examinations usually take about two to three weeks. The default timeslot is from Monday to Saturday with two different sessions, which are morning and afternoon sessions. Each session will have a group of courses sit for the examination for three hours.

3.3 Modelling and Formulation

In the final examination timetabling problem for this faculty, the following notations are used in our mathematical modelling.

3.3.1 Parameters

$n =$ number of courses

$g =$ number of course groups

$m =$ number of rooms

$d =$ number of days

$t =$ number of timeslots

3.3.2 Sets

C: set of courses $= \{c_1, c_2, \dots, c_n\}$

H: set of course groups $= \{h_1, h_2, \dots\ h_g\}$

R: set of rooms $= \{r_1, r_2, \dots, r_m\}$

T: set of timeslots $= \{t_{11}, t_{12}, \dots, t_{dt}\}$, where $dt =$ day and timeslot

S_{C_i}: student set of course i, where $i = 1, 2, \dots, n$

Set C is a set of courses which is grouped into different sets of course groups G. Course group g_j is searched for available timeslot t_{dt}. All the courses in the course group is allocated into the most fitted room r_m.

3.3.3 Constraints

There are two types of constraints in the scheduling problem. Hard constraints (HC) are those which must be satisfied in order to get a feasible solution. Fulfilling the soft constraints (SC) is optional but the satisfaction of soft constraints is to ensure a better-quality solution.

HC_1 A student can only have one examination at a time.

HC_2 Room capacity must fit in the size of allocated exams.

SC_1 Minimise the total room capacity wastage.

SC_2 Minimise the number of student sectioning in a course.

SC_3 Minimise the number of continuous examination.

3.3.4 Multi-Objective

In this study, the faculty timetabling can be modelled as a multi-objective optimisation problem, where the multi-objectives are as follow:

(a) Maximise the room utilisation

(b) Minimise student sectioning for each course

(c) Minimise the number of continuous examinations

(d) Fulfil all the priority courses during the examination period.

3.4 Conclusion

The faculty final examination timetable case study has been discussed in this chapter. Sets and parameters, constraints and objective functions were all defined.

The current timetabling practice in FCSIT produces a clash-free timetable but it is not a good quality solution. The venue resources are not being utilised as much as possible. To cope with more practical constraints, the wastage of room capacity is too high in this solution. When the students from a course is split into many rooms, it will lead to an increase in invigilators on duty. The way of doing the student sectioning for each course is important to minimise the room wastage. Besides that, the consideration of priority courses and continuous examinations are also important when doing the scheduling. All the priority courses will need to sit for the examination first. At the same time, the students cannot have continuous examinations. The consideration of these two factors increases the challenge of proposing a good quality final examination timetable.

CHAPTER 4

TWO-STAGE HEURISTIC

4.1 Introduction

This chapter discusses the proposed heuristic method for solving the examination timetabling problem. The two-stage heuristic method will be explained in detail. The discussion begins with the algorithm of the two-stage heuristic, which are course grouping and timeslot-room allocation.

4.2 Two-Stage Heuristic Method

As discussed in the previous chapter, the final examination timetabling problem is an NP-hard problem (Abdul-Rahman et al., 2014). With the extra constraints on the course, student and room, the problem has become more complicated. Besides that, the problem size in FCSIT has increased from 2016 to 2018 as illustrated in Figure 4.1.

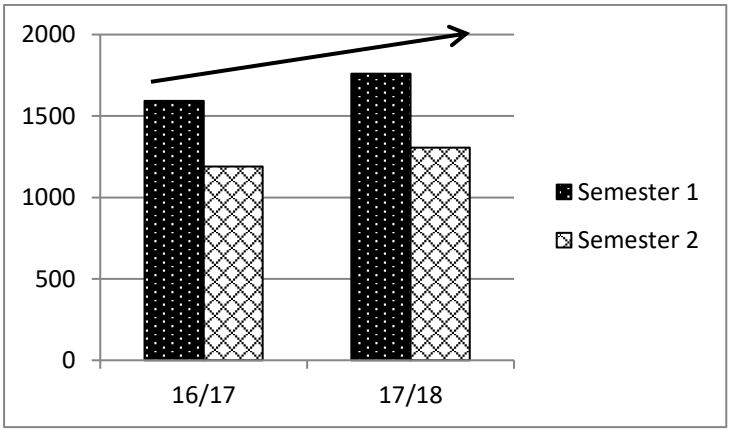


Figure 4.1: Increment in number of students from 2016 to 2018

Therefore, the two-stage heuristic method is needed to solve the timetabling problem. The two stages are course grouping and timeslot-room allocation (shown in Figure 4.2).

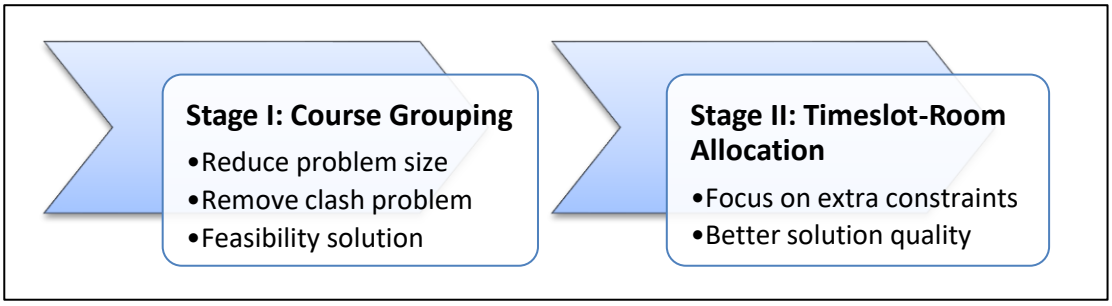


Figure 4.2: Two-stage heuristic algorithm

In the course grouping stage, all the courses will be grouped into a number of groups. The purpose of course grouping is to produce a feasible final examination timetable. All the courses are grouped based on the student matric number. The main criteria of the grouping is that there is no common student in each group. Indirectly, this can overcome the issue of the repeated student. The grouping is to ensure that all the students do not have a clash in their examinations. The course group created can also reduce the problem size. This is because the number of groups created represents the number of timeslots needed for the examination period. Therefore, a smaller number of groups is highly recommended in order to shorten the examination period.

In the second stage, the quality solution will be focused during the timeslot-room allocation. All the course groups created in the first stage will be distributed into the available timeslots. The constraints of *priority course* and students' continuous exams will be considered during the distribution. The *priority course* will take the examination as early as possible. Meanwhile, continuous exams must be prevented in order to generate a good quality solution.

After all the course groups have their own timeslots, the courses in each group will be allocated into suitable rooms for the examination. Student sectioning will be included when there is not enough capacity for a course. Similarly, the combination of multiple courses in a room will also be included to minimise room wastage. Proper student sectioning and the combination of multiple courses can help to maximise the room utilisation. Thus, the maximisation of room utilisation with an efficient selection of rooms and timeslots can produce a good quality solution.

4.3 Stage I: Course Grouping

The problem scale will be reduced by setting n courses grouped into g course groups, where $n < g$. The most important criterion in the course grouping is each student in a course group can only take a maximum of one course. This is important in order to ensure the feasibility of the solution, which is the no clash issue in the generated timetable. During the course grouping, there are a few important rules that need to be followed, which are:

Rule 1: Pairing Ability

The *pairing ability* (as illustrated in equation (1)) of each course will be determined. The *pairing ability* of a course is the total number of courses that can be grouped with a particular course.

$$\textit{Pairing ability of course, } P_{C_i} = \sum_{j=1}^{n} X_{ij} \text{ where } X_{ij} = \begin{cases} 0, & S_{C_i} \cap S_{C_j} \neq \emptyset \\ 1, & S_{C_i} \cap S_{C_j} = \emptyset \end{cases} \quad (4.1)$$

The equation above shows the way to get the *pairing ability* of course i, P_{C_i}. If there are no common students between two courses ($S_{C_i} \cap S_{C_j} \neq \emptyset$), then the P_{C_i} will

increase by 1, else the P_{C_i} will remain the same. Courses with a low *pairing ability* may have a lesser chance to be grouped with the other courses. Therefore, those courses will be the first consideration during the grouping. Figure 4.3 demonstrates the calculation of the *pairing ability* for each course.

Given set of student in each course, S_{C_1}, S_{C_2} and S_{C_3}:

$$S_{C_1} = \{A, B, C, D\}$$

$$S_{C_2} = \{E, F, G\}$$

$$S_{C_3} = \{A, B, E, F\}$$

Thus, the *pairing ability* of each course are as follows:

$P_{C_1} = X_{11} + X_{12} + X_{13}$	$P_{C_2} = X_{21} + X_{22} + X_{23}$	$P_{C_3} = X_{31} + X_{32} + X_{33}$
$= 0 + 1 + 0$	$= 1 + 0 + 0$	$= 0 + 0 + 0$
$= 1$	$= 1$	$= 0$

Figure 4.3: Calculation of *pairing ability* of each course

In Figure 4.3, the *pairing ability* for each course C_1 and C_2 is 1, which means there is only one course that has no common student with course C_1 and C_2. While, the *pairing ability* for course C_3 is 0. This means that C_3 has common students with both C_1 and C_2.

Rule 2: Priority Course

Priority course means the course that is given *priority* to have an earlier examination date compared to the other courses. The *priority course* is an important constraint during the course grouping. This is because all the *priority courses* need to be grouped together or even

into a few groups only. This is to make sure that all the *priority courses* will not be scattered among all the groups created. Therefore, those courses with *priority* are to be grouped first compared to the courses with no *priority*.

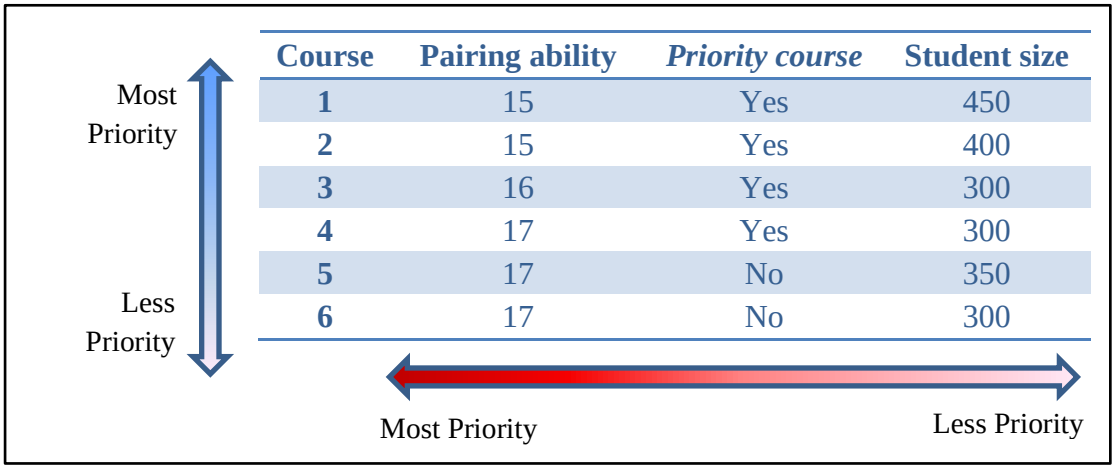
Rule 3: Student Size

Before the course grouping, the student size of each course will be determined. Courses with a large student size will have a high *priority* for grouping. This is because the lecturer of these courses need more time to mark the examination papers.

Since the total number of course groups created represents the total number of timeslots needed for the final examination, all the three rules stated above must be followed. Therefore, there are two main steps included in this stage, which are:

Step 1: Course sorting

As shown in Figure 4.4, the list of courses will be sorted based on the three different rules and orders.



Course	Pairing ability	*Priority course*	Student size
1	15	Yes	450
2	15	Yes	400
3	16	Yes	300
4	17	Yes	300
5	17	No	350
6	17	No	300

Figure 4.4: List of sorted courses

Courses with the criteria of low *pairing ability*, with *priority* status and large student size have high *priority* during grouping. If the *pairing ability* of the courses are tied, the *priority course* will be grouped first. Meanwhile, if there are a few courses with *priority* status, the courses with a large student size will have a higher *priority* during the course grouping.

Step 2: Assign courses into group

After all courses are sorted based on the three rules as stated before, the process of assigning courses into available groups will be started. All the course assignment to course groups will follow the sorted course sequence.

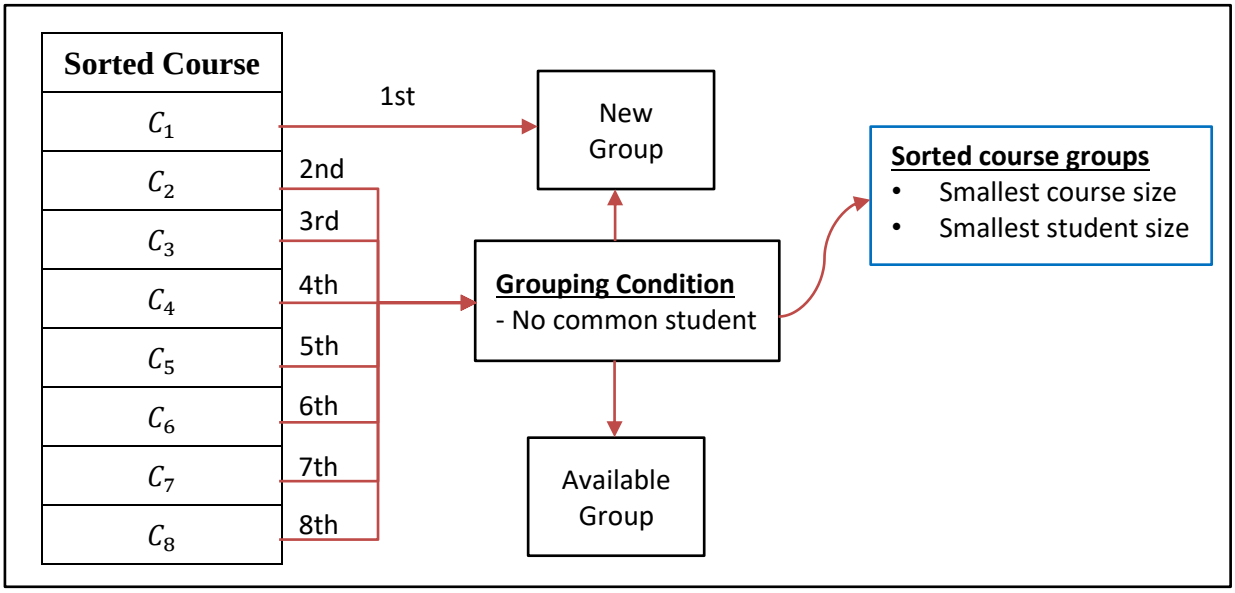


Figure 4.5: Assign course into available group

As shown in Figure 4.5, the first course will be automatically assigned into the first created group. Then, the second course, C_2 will check through available course groups, based on the following criteria:

Step i: Choose the course group with no common student in the available course group i.

Step ii: If more than one course group are available, choose the course group with the least number of courses.

Step iii: If tied, choose the course group with the least number of students.

Step iv: Assign to new course group j.

The group will be sorted by the course size in an ascending order. Similarly, the group will also be sorted by student number in an ascending order. This is in order to make sure that all the course groups have a balanced number of courses and students.

Next, the repeated process of searching for available groups will be carried out for all the courses with no groups. There is one most important condition during the grouping process, which is there be no common student among the courses in a group. This is to make sure that each student will only have one exam at a time. Thus, in each course, the student identification number is used to check for the condition. If there is a group has no common student with the course, then that course will be assigned into the group. However, if there is no available group, a new group will be created for that course.

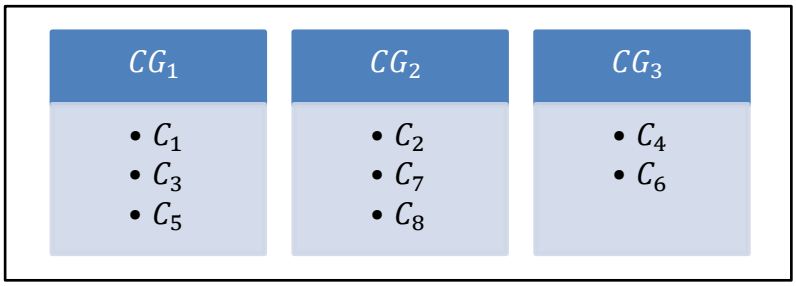


Figure 4.6: List of created course groups

By the end of stage I, a list of course groups are created as in Figure 4.6. By using students' matric numbers, there will be no common student between the courses in each course group.

4.4 Stage II: Timeslot-Room Allocation

After the course grouping and distribution of groups in different timeslots, the courses in each group will be allocated into the available rooms. There are three phases in the timeslot-room allocation stage, which are:

4.4.1 Pre-allocation Phase: Group Sorting

The course groups created in Stage I will be assigned to the available timeslots. All the groups created in Stage I will be sorted based on *priority course* and student size.

- The number of *priority course*: The group with a higher total number of *priority course* will be the first considered when assigning groups into timeslots. This is to make sure all the *priority* courses take the examinations as early as possible.
- Student size: All the groups will be sorted based on the total student size in descending order. The group with a larger student size has the priority to be allocated into the earlier timeslots. This will allow all the lecturers of the courses in the group to have more time to mark the examination papers.

4.4.2 Timeslot Allocation Phase

Before the assigning of course groups into the available timeslots, all the available timeslots will be sorted in an ascending order of time. Then, the timeslot assigning will start with the highest priority group, g_j (shown in Figure 4.7). All the course groups will go through the iterative process of searching an available timeslot.

High Priority

Low Priority

Group	No. of *priority course*	Student Size
2	6	900
3	6	890
1	5	900
4	5	900
5	5	800
7	0	890
6	0	890

Figure 4.7: List of course groups

Figure 4.8 shows the details of assigning each timeslot to a suitable course group. In step (a), the first course group will be automatically assigned to the first timeslot. Then, in steps (b) and (c), when assigning the next timeslot, the next x of the top sorted group will be used to calculate the penalty with the group in the previous timeslot. The x can be any value which can be set by user, where x is larger than 1. The purpose of getting the penalty between groups is to check for student continuous examination. The algorithm to avoid continuous examinations is as below:

i. An examination timeslot is determined.

ii. Among the top x course groups, the top x groups in the sorted list will be checked for penalty on continuous examinations, where x can be any number which can be set by the user.

iii. By comparing with the previous examination timeslot, all the student cohort of courses between two different groups (previous group, g_p and the potential course group to be assigned, g_a) will be determined.

iv. The two courses between g_p and g_a that have the same student cohort will have one penalty added on the g_a. Meanwhile, the penalty of the g_a will remain the same if the student cohort of courses is different.

v. Steps (iii) and (iv) will be looped continuously until all the penalties of x groups are determined.

vi. The group that has the lowest penalty will be directly assigned to the examination timeslot.

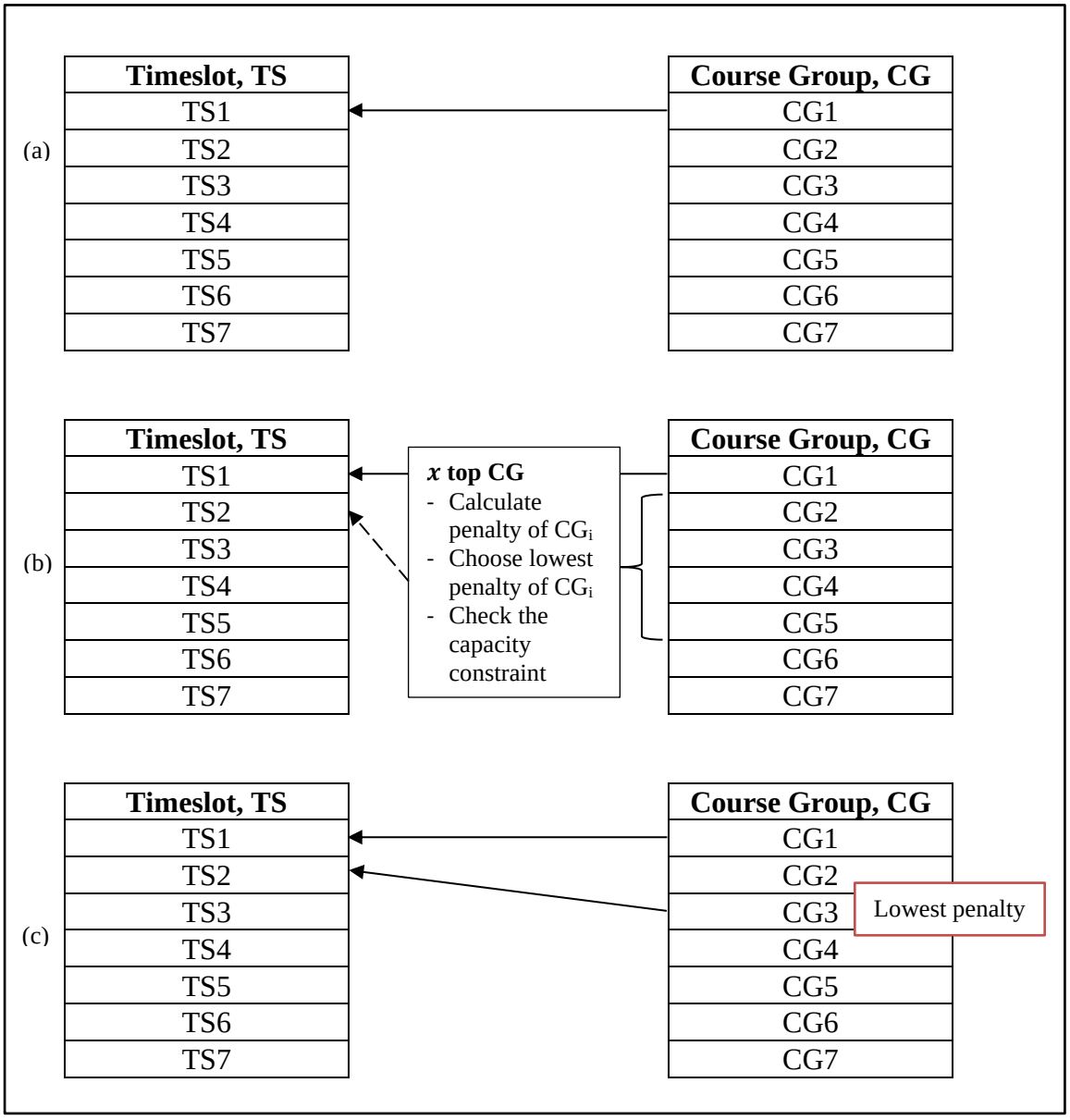


Figure 4.8: Assigning timeslot to course group

Algorithm 4.1 shows the process of searching for the most suitable course group timeslot.

Algorithm 4.1: Search most suitable course group into timeslot

```
g_t = temporary group
g_t = NULL
FOR j = 1 to x
    penalty = 0
    FOR every course i in g_p
        FOR every course j in g_x
            IF programme and semester taken of course i
                                = programme and semester taken of course j
                penalty = penalty + 1
            ENDIF
        ENDFOR
    ENDFOR
    IF g_t = NULL
        g_t = g_x
    ELSE
        IF penalty of g_t < penalty of g_x
            g_t = g_x
        ENDIF
    ENDIF
ENDFOR
g_t = selected group for the timeslot
```

After getting the lowest penalty between the groups, the group will be checked whether it fits into the timeslot. If the group does not fit into the timeslot, all the unallocated groups will be regrouped where the maximum size of the group must not be greater than the size of the timeslot. However, if the group size fits into the timeslot capacity, then the group will be automatically assigned into the timeslot. All the unallocated course groups will iteratively go through the process of penalty calculation and checking room capacity. Finally, each group will have their own timeslot.

4.4.3 Room Allocation Phase

After all the groups have been allocated their own examination timeslot, the distribution of courses into rooms is the next step for this stage. This is an important step to achieve the objectives of this research. The objectives include maximising room utilisation and minimising the number of splitting into different rooms for each course.

All the courses will be allocated into the available examination rooms. In each group, the course that has set a particular examination room will be the first to search for examination rooms. This is to make sure that all the particular courses can be allocated into the particular rooms.

At the same time, all the courses in each group will be sorted by the total number of students in a descending order. The course with a larger number of students will be allocated an available examination room. This allocation of courses can minimise the number of student sectioning into several rooms. As the number of student sectioning is lower, the number of invigilators on duty will also be smaller.

There are different cases when each course i searches for an examination room, which are:

Case I: A room fits all the students, but $i \neq n$, where n is last course in the group

- If the partially used room is available, then course i will be set into the room.
- Else, course i will be set into the largest room.

Case II: A room fits all the students, but $i = n$, where n is last course in the group

- If the partially used room is available, then course i will be set into the room.
- Else, course i will be set into the smallest room.

Case III: Do not fit all the students

- The course i will search for the largest room and the remaining will search for a room with the best fit.

Figure 4.9 shows the example of allocation in a group into the rooms in an examination timeslot. In each group, all the courses will be first sorted based on their total number of students.

- Figure 4.9 (a): Course TMF2234 falls in Case III which will be split into several rooms.
- Figure 4.9 (b): Course TMN3053 falls in Case I and will find the largest room.
- Figure 4.9 (c): Course TMP3414 falls in Case I and will find the room that was used before.
- Figure 4.9 (d): Course TMP3613 falls in Case II and will find the smallest and fitted room.

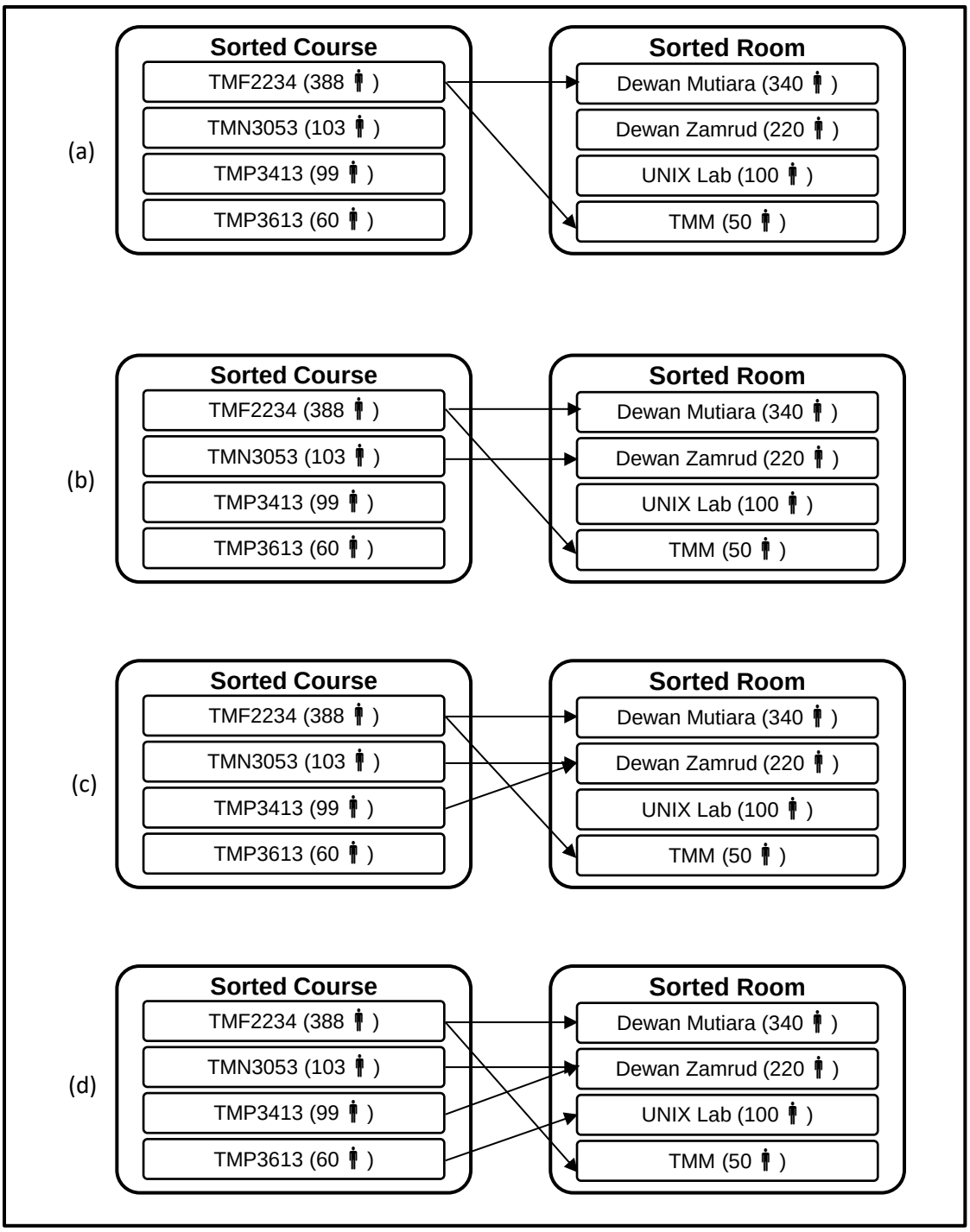


Figure 4.9: Allocation of courses with student sectioning into available rooms

4.5 Conclusion

A two-stage heuristic method is proposed to solve the faculty final examination timetabling problem. Figure 4.10 shows the overall process of the two-stage heuristic method. In the first stage, the course grouping will group the courses with no common students into different groups of courses. This stage can decrease the complexity of the process in the second stage, which is the timeslot-room allocation. In the second stage, all the course groups will be distributed into the available timeslots and assigned to the rooms for each course.

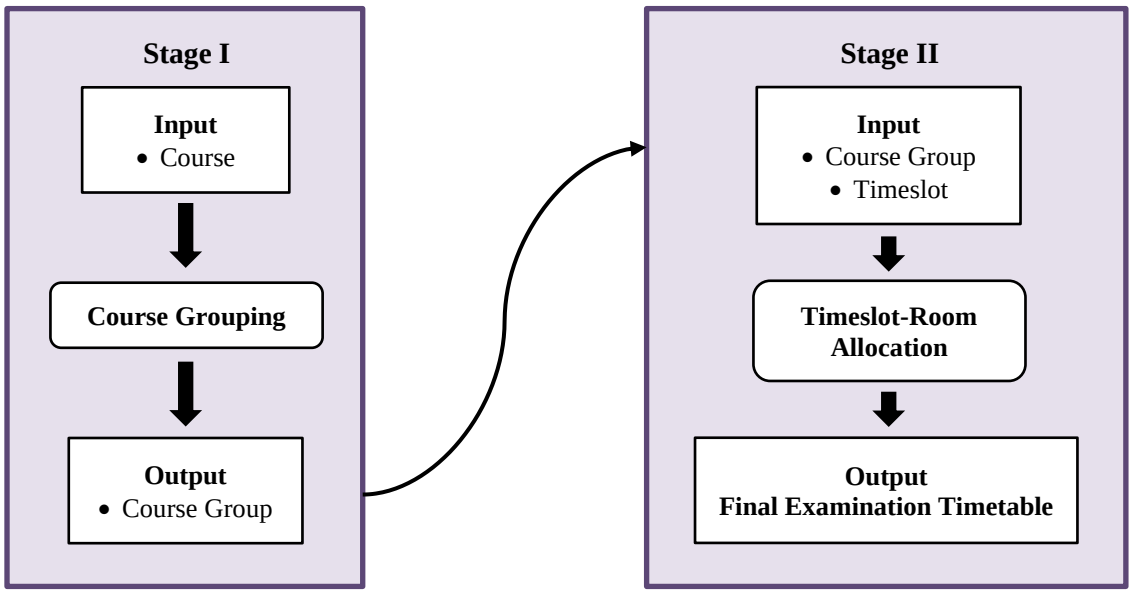


Figure 4.10: Overall process of two-stage heuristic method

Student sectioning is considered in this study to minimise the number of splitting for each course. A low number of student sectioning means that fewer invigilators would be needed. Meanwhile, the combination of different courses into a room can also maximise room utilisation. This reduces the number of invigilators on duty and minimises room wastage during the final examination.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Introduction

In this chapter, the two-stage heuristic algorithm is evaluated and compared with a range of datasets with different settings. The algorithm is developed in a simulation model, which is called the Final Examination Scheduling System 2.0 (FESS 2.0). FESS 2.0 is an enhancement of FESS 1.0, where both use the two-stage heuristic algorithm. Thus, FESS 1.0 (refer Chapter 1, page 3) is included in this analysis. There are a few datasets from the faculties in UNIMAS that are used to validate the improvement from FESS 1.0 to FESS 2.0.

5.2 Dataset Description

Four semester datasets from FCSIT were collected to validate and evaluate the proposed algorithm. The dataset for each semester is defined as 16/17-1 for 'Year 2016/2017, Semester 1', 16/17-2 for 'Year 2016/2017, Semester 2' and so on. Table 5.1 shows the general information of the datasets from FCSIT.

Table 5.1: Datasets general information in FCSIT

Dataset	**16/17-1**	**16/17-2**	**17/18-1**	**17/18-2**
Course	51	48	52	49
Student	1593	1189	1760	1306
Course-Student Enrolment	6335	4240	6840	4403
Exam Period (day)	11	12	11	11

Generally, FCSIT has an average of 50 courses every semester. During Semester 2, the student size is slightly smaller because most of the third-year students are away for a semester of internship. As the student size becomes smaller, the course-student enrolment also decreases.

In order to validate FESS 2.0 further, experiments were also conducted on other faculties in UNIMAS. These faculties included the Faculty of Social Sciences (FSS), Faculty of Economics and Business (FEB), Faculty of Engineering (FENG) and Faculty of Cognitive Sciences and Human Development (FCSHD). Table 5.2 shows the general information of datasets from those faculties.

Table 5.2: Datasets from faculties

Dataset	**16/17-1**	**16/17-2**			**17/18-1**	**17/18/2**
Faculty	FSS	FEB	FENG	FCSHD	FEB	FEB
Course	62	70	90	43	53	76
Student	2038	1853	1614	1600	1838	1893
Course-Student Enrolment	8163	7914	5051	6406	7270	7586
Exam Period (day)	11	12	12	12	11	11

Among the faculties, FENG had the highest number of courses with the lowest number of students enrolled. Each course in FENG had an average of 56 students. In other faculties, most of the courses on average had more than 100 students. Therefore, FENG had the highest number of course-student enrolment when compared with FSS, FEB and FCSHD.

Based on the datasets collected, there were a few courses in FEB and FCSHD that had a large size of course-student enrolment with more than 400 students. As stated in Chapter 3, the largest room capacity for the final examination is only 400. Therefore, the final examination for these courses must be conducted in multiple exam venues.

5.3 FCSIT Timetabling Practices

In FCSIT, the final examination timetabling was scheduled manually by the faculty administrator before Semester 2014/2015-2. The faculty administrator usually took about two weeks to generate a final examination timetable. Even after the timetable had been generated, the clash problem still existed.

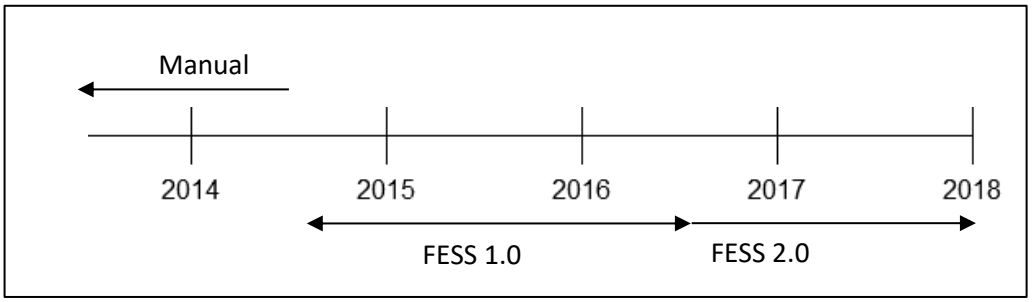


Figure 5.1: Examination timetabling practices in FCSIT

As shown in Figure 5.1, since 2014/2015-2, FCSIT has started to implement FESS 1.0 with the two-stage heuristic algorithm. FESS 1.0 can quickly generate a feasible final examination timetable with no clashes. There are some limitations to FESS 1.0 such as utilised room allocation, high number of student sectioning for each course, continuous examinations and the existence of empty slots within the examination period. Therefore, starting from 2016/2017-1, FCSIT began to use FESS 2.0 to generate the timetable. FESS 2.0 is the enhanced version of FESS 1.0 where FESS 2.0 is implemented with the two-stage heuristic algorithm that is applied in this research. It can overcome the limitations in FESS 1.0 such as room utilisation, student sectioning, continuous examinations and the empty slots. The next section is going to discuss the overall comparison between FESS 1.0 and FESS 2.0.

5.4 Computational Analysis

The FESS 2.0 was coded in PHP web-based programming language. All experiments were carried out in 2.40GHz on an Intel® Core i5 processor with 4GB RAM and MySQL database.

5.4.1 Overall Solutions Comparison

Figure 5.2 presents the overall solution results for both FESS 1.0 and FESS 2.0. Both solutions are able to generate a timetable in less than a minute. This is due to the heuristic simplicity in the algorithm. However, there is still a slightly shorter computing time in FESS 2.0.

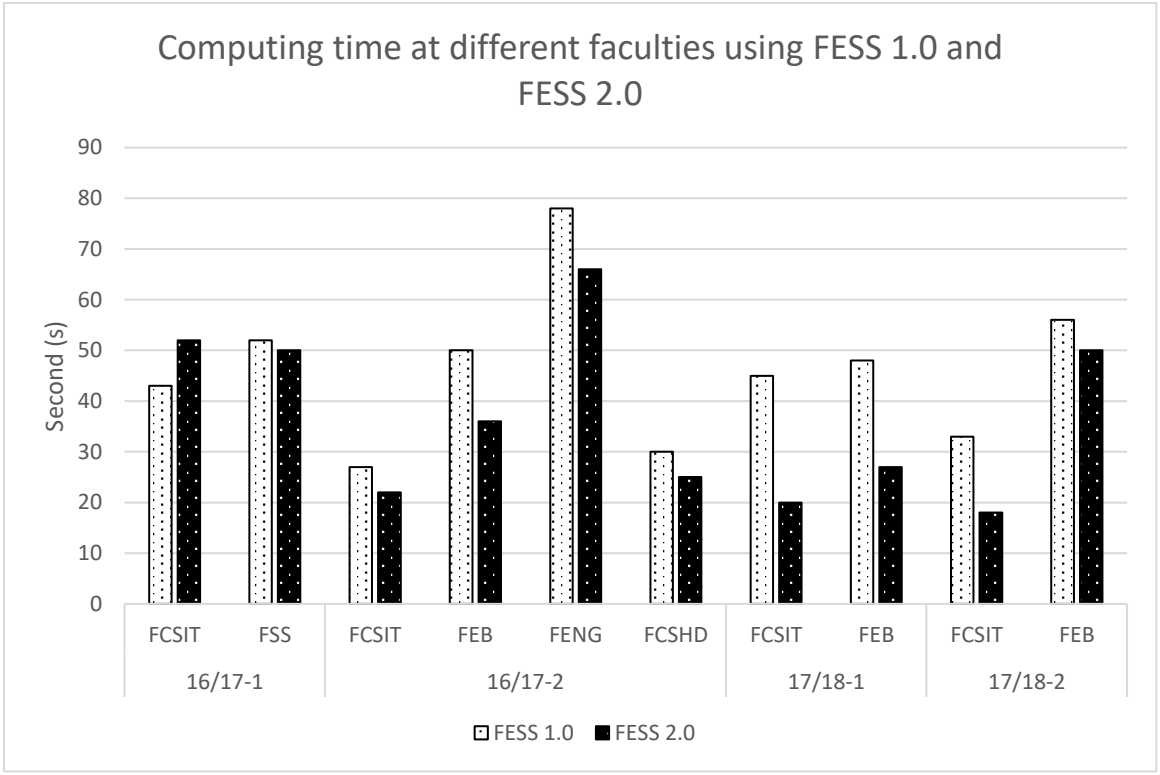


Figure 5.2: Computing time at different faculties using FESS 1.0 and FESS 2.0

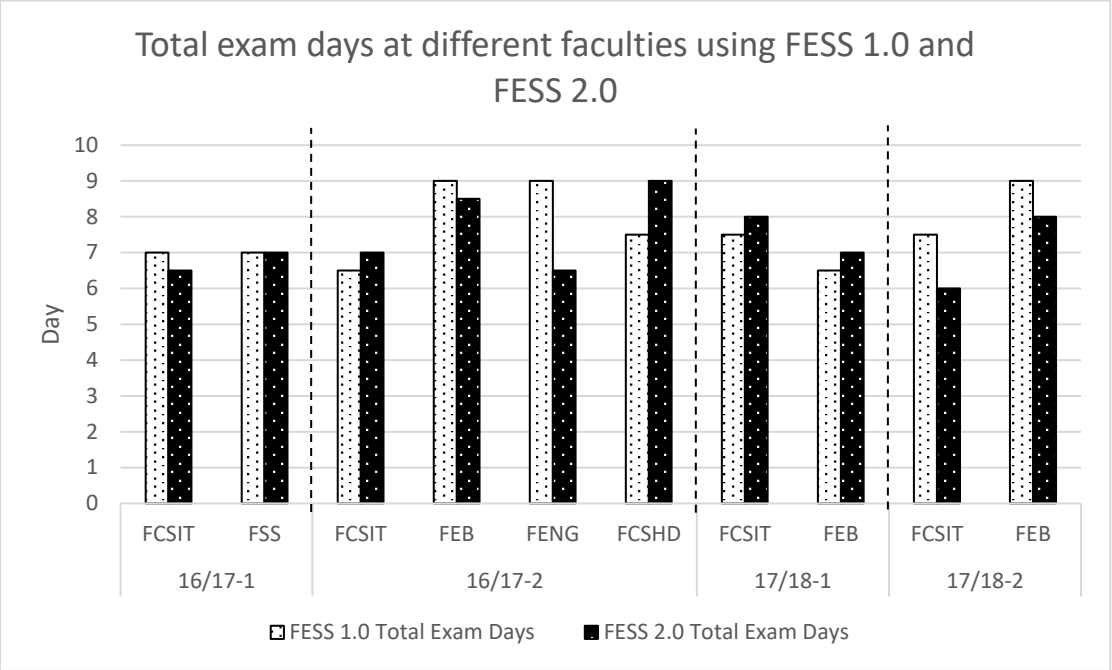


Figure 5.3: Total exam days at different faculties using FESS 1.0 and FESS 2.0

As shown in Figure 5.3, all the datasets from different faculties processed in the FESS 1.0 and FESS 2.0 generated a timetable with less than 10 days of final examination. The generated timetable in FESS 1.0 is longer than the timetable generated in FESS 2.0. This is because more operational constraints are considered in FESS 2.0, such as course priority and balanced number of courses in all groups. All the courses with priority need to have the examination as early as possible. Therefore, during the course grouping, the courses with priority need to be grouped first.

The total exam days used in the examination is based on the total number of groups created in Stage 1.0. In FESS 2.0, all the groups created have a balanced number of courses. Course balancing can prevent a lot of student sectioning in a course. As there are many large-sized courses in the same group, this means that student sectioning is required in Stage 2.0 of timeslot-room allocation. Therefore, in order to avoid student sectioning, a balanced number of courses in a group needs to be considered in the course grouping.

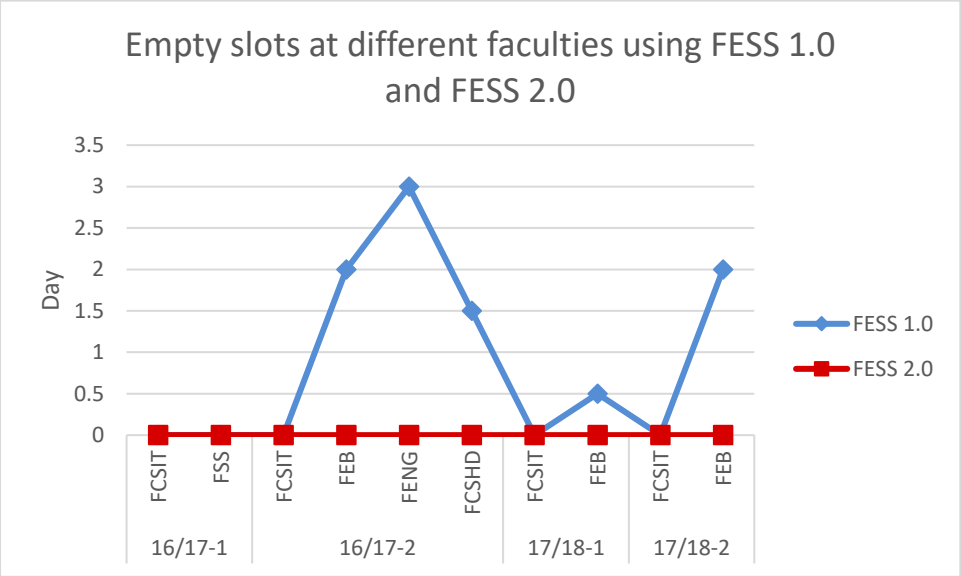


Figure 5.4: Empty slots at different faculties using FESS 1.0 and FESS 2.0

As shown in Figure 5.4, even though the total exam days generated in FESS 1.0 are shorter than FESS 2.0, there are a few timetables generated in FESS 1.0 which consist of empty slots during the final examination period. An empty slot means one session does not have any examination between two sessions. Therefore, the empty slot is not practical in a feasible solution. If an empty slot is present, the user has to manually edit the timetable to move some of the examination courses to fit the empty slot. Therefore, the algorithm in FESS 2.0 has solved the empty slot problem. There are no more empty slots during the final examination period.

5.4.2 Room Utilisation

During the final examination, room utilisation is also an important consideration. The total number of rooms used may affect the number of invigilators on duty. When the total number of rooms used increases, it means that more invigilators are needed to supervise the

examination. Meanwhile, the total unutilised room capacity may also affect the room utilisation constraints. A large unutilised room capacity will increase room wastage during the final examination. This is not a favourable solution in real life. Therefore, in FESS 2.0, maximising room utilisation is an important objective when scheduling the final examination timetable.

There are two ways to maximise the room utilisation i.e. the right examination room size and pack more courses in the examination rooms. The right size of examination room means the room capacity that is most fitted to the student number in the course. Furthermore, a room that has been allocated with a course will also be the first choice to be allocated to other courses. Therefore, the room that has been used and the most fitted size of room will be the first consideration during the allocation of courses into available rooms.

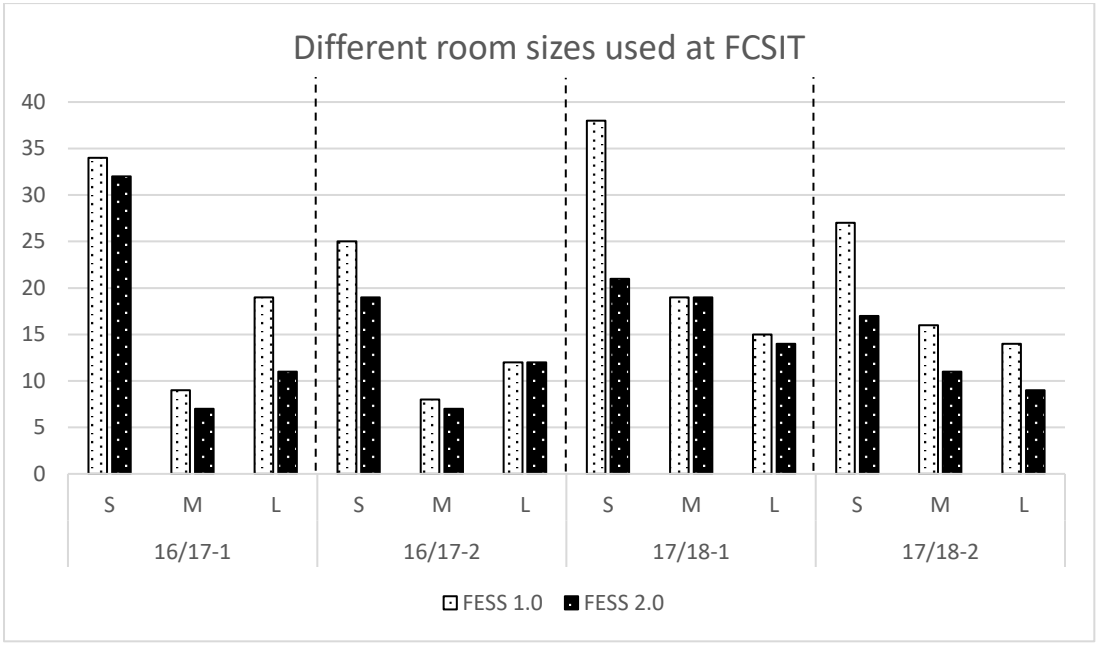


Figure 5.5: Different room sizes used at FCSIT

Table 5.3: Room utilisation at FCSIT

Dataset	**16/17-1**		**16/17-2**		**17/18-1**		**17/18-2**	
FESS	**1.0**	**2.0**	**1.0**	**2.0**	**1.0**	**2.0**	**1.0**	**2.0**
Total Room Used	62	50	45	38	72	54	57	37
Improvement (%)	-19%		-15.5%		-25%		-35.09%	
Room Capacity Used	6335		4240		6840		4403	
Unutilised Room Capacity	2245	625	2138	1200	2572	770	3603	577
Improvement (%)	-72%		-43.87%		-70.06%		-83.99%	

Figure 5.5 and Table 5.3 show the room utilisation during the final examination at FCSIT. Since most of the third-year students went for their internship during Semester 2, the total number of rooms used during this semester was much fewer compared to Semester 1. The total number of rooms used in the generated timetable by FESS 2.0 decreased more than 15% compared with the generated result using FESS 1.0. Similarly, the unutilised room capacity also decreased when using the FESS 2.0 solution. The unutilised room capacity means the capacity left in the room that is used during the final examination. The unutilised room capacity has decreased by up to 83.99%; this can reduce room wastage during the final examination.

Table 5.4: Room utilisation at FEB

Dataset	**16/17-2**		**17/18-1**		**17/18-2**	
FESS	**1.0**	**2.0**	**1.0**	**2.0**	**1.0**	**2.0**
Total Room Used	75	65	61	52	70	61
Improvement (%)	-13.33%		-14.75%		-12.86%	
Room Capacity Used	8178		7270		7927	
Unutilised Room Capacity	2386	1192	1464	610	2698	843
Improvement (%)	-50.04%		-58.33%		-68.75%	

At FEB, more than 7000 seats or room capacity are needed for the final examination in each semester. Therefore, the total number of rooms used at FEB is higher than at FCSIT. As shown in Table 5.4, the total rooms used generated in FESS 2.0 for each semester

decreased more than 12% or on average 10 rooms fewer. Consequently, the total unutilised room capacity also decreased. In the timetable scheduled by FESS 2.0, the unutilised room capacity went down more than 50%. For example, during Semester 2 2017/2018, the unutilised room capacity decreased from 2698 to 843, which is about 68.75%.

Table 5.5: Room utilisation at other faculties

Dataset	**FSS, 16/17-1**		**FENG, 16/17-2**		**FCSHD, 16/17-1**	
FESS	**1.0**	**2.0**	**1.0**	**2.0**	**1.0**	**2.0**
Total Room Used	63	58	66	58	61	58
Improvement (%)	-7.94%		-12.12%		-4.91%	
Room Capacity Used	8163		5754		6406	
Unutilised Room Capacity	1557	852	2018	518	879	839
Improvement	-45.28%		-74.33%		-4.55%	

Table 5.5 shows the room utilisation at FSS, FENG and FCSHD, in different semesters of final examinations. Even though all the datasets are from different faculties and different semesters, the total rooms used were reduced by at least 4.91%. Since the student number in each course in FSS and FCSHD is mostly larger than the courses in FENG, the improvement in total rooms used in both faculties also became lower. Consequently, the unutilised room capacity in FENG decreased by up to 1500. Meanwhile, the total rooms used in FSS and FCSHD were reduced by about 45.28% and 4.55%, respectively. The reason for the lower improvement in FCSHD is because the examination rooms owned by the faculty are very small. The faculty only owns one room that can fit 60 students and another four rooms have a capacity of 15 students in each room. Therefore, the decrease in the unutilised room capacity in FCSHD is not as big as the other faculties.

5.4.3 Student Sectioning by Course

The minimising of student sectioning for each course is an important task when scheduling the final examination timetable. The number of rooms used for a course will affect the number of invigilators on duty. If the course uses a higher number of rooms for the examination, this means that the invigilators on duty will also increase. According to Figure 5.6 and 5.7, FESS 1.0 in Semester 1 2016/2017 at FCSIT had 2 courses using 7 rooms each. This means that 14 rooms were needed for the examination. This is an impractical solution as it may require more invigilators during the examination. Therefore, Figure 5.7 to Figure 5.9 show the graphs of the student sectioning analysis on the datasets from different faculties in UNIMAS by using FESS 1.0 and FESS 2.0.

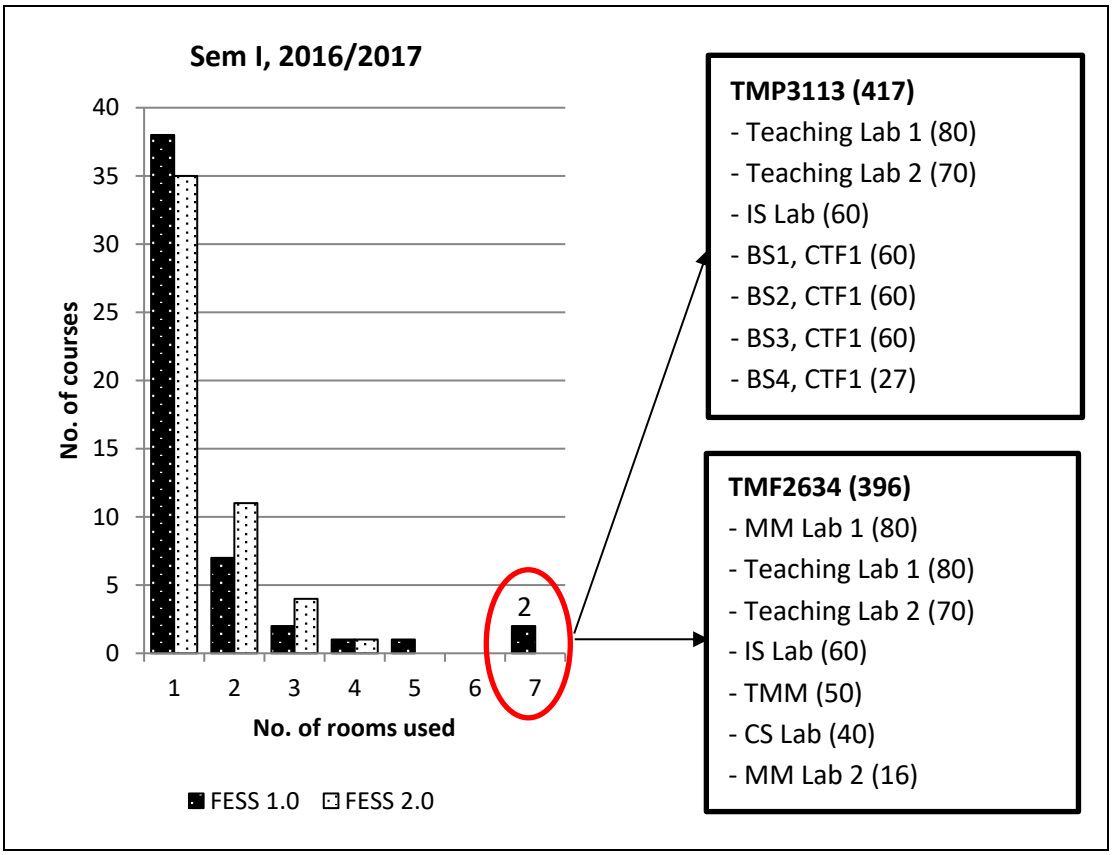


Figure 5.6: Student sectioning by course at FCSIT in Semester 1, 2016/2017

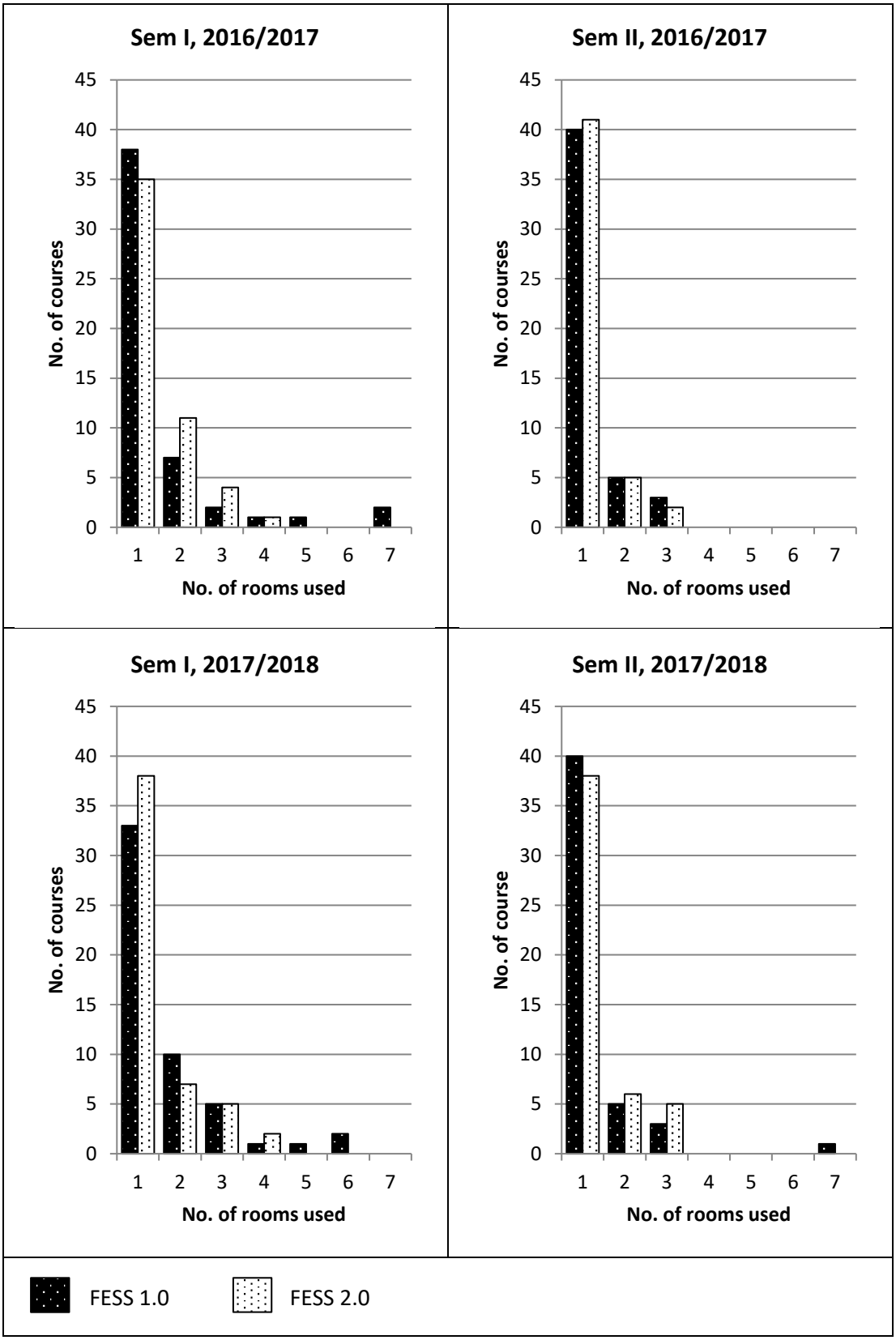


Figure 5.7: Number of rooms used based on student sectioning algorithm at FCSIT

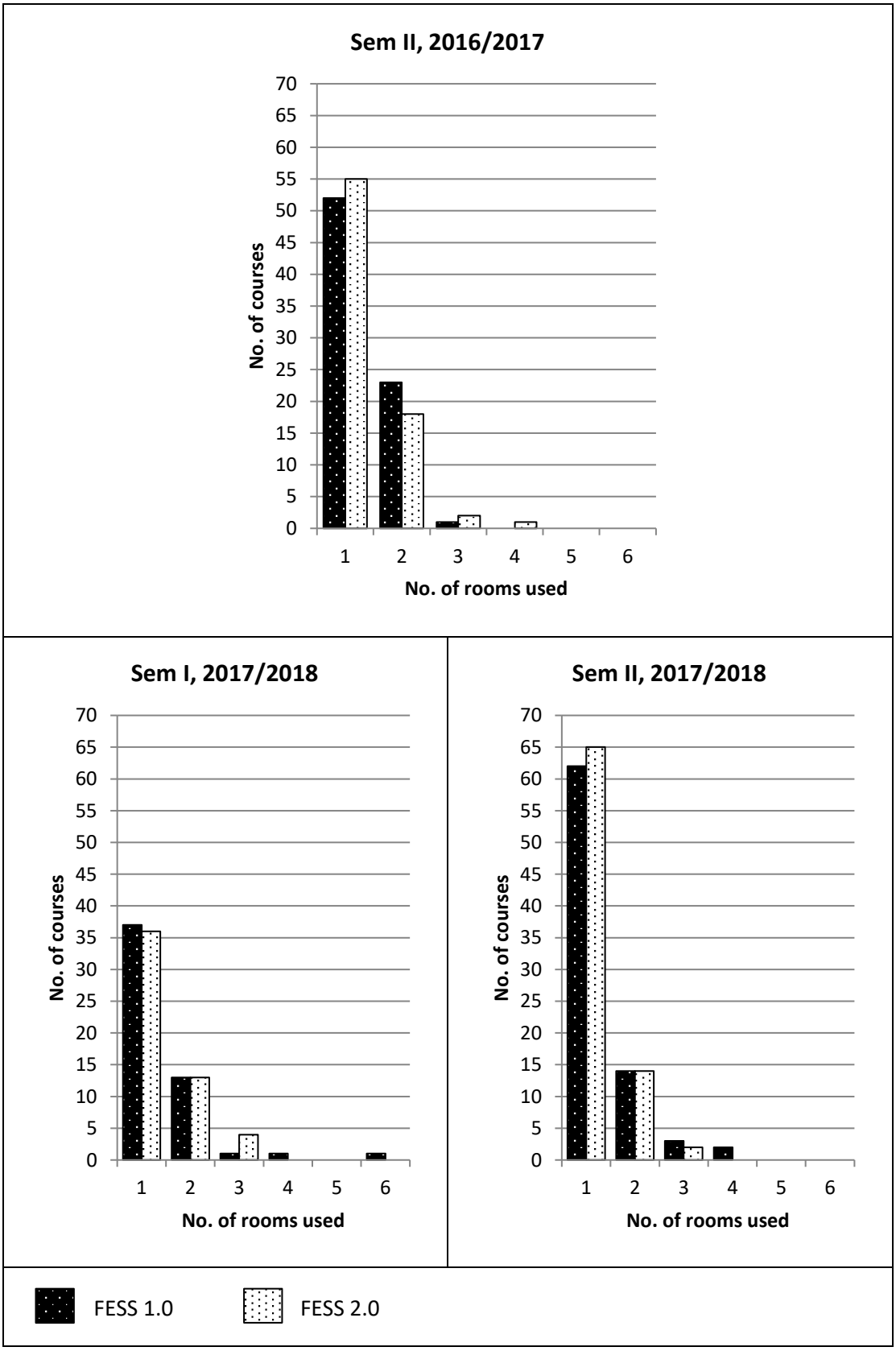


Figure 5.8: Number of rooms used based on student sectioning algorithm at FEB

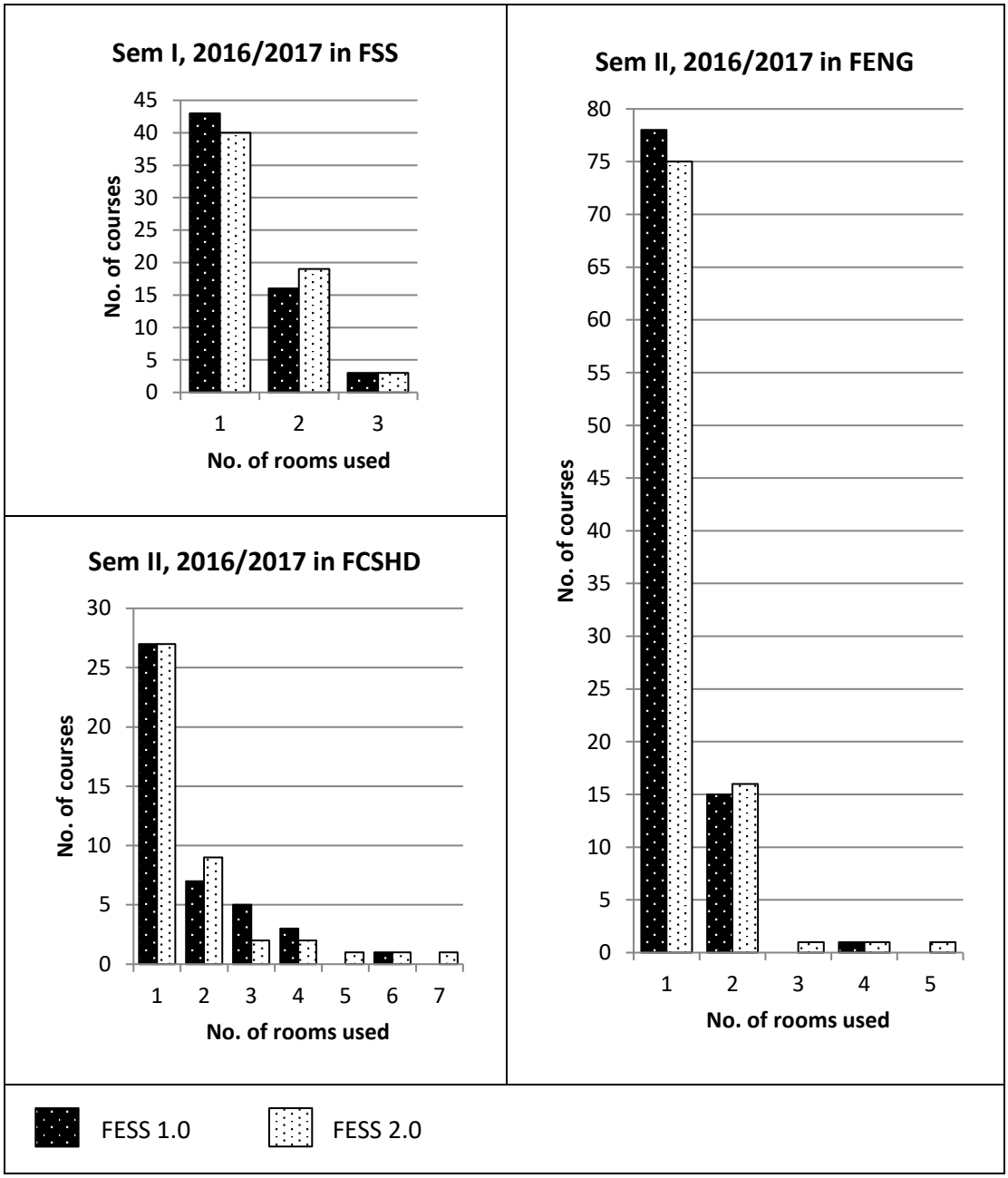


Figure 5.9: Number of rooms used based on student sectioning algorithm at other faculties

Overall, the student sectioning for each course improved when using FESS 2.0 compared to FESS 1.0. Most of the timetables generated in FESS 1.0 may use up to 7 rooms. However, the minimising of student sectioning is practical for FESS 2.0 to achieve. FESS 1.0 uses the greedy packing heuristic, where it always chooses the biggest room capacity during the room allocation for each course. FESS 2.0 is applied with some rules during the room allocation process (refer to Chapter 4, page 32). Therefore, the solution of FESS 2.0 produced a better result based on the constraints of student sectioning, whereby the number of rooms used was only up to 4.

Similarly, the datasets from FEB and FSS also showed improvement when using FESS 2.0. As shown in Figures 5.8 and Figure 5.9, student sectioning in a course during Semester 2 2016/2017 at FEB, FENG and FCSHD did not improve. This is because in FESS 2.0, a few course groups consisted of more than one course with a large student size. As more courses have a large student size, then more large-sized rooms are needed for the final examination. Therefore, the higher number of rooms used in a course cannot be prevented if total number of large course sizes is more than the total number of large rooms.

5.4.4 Timetabling Solution Quality

Stage I: Course Grouping

Course grouping is an important stage in the scheduling of the final examination timetable. The result may be affected by the course grouping. Table 5.6 shows the different conditions of course groups that affect the results.

Table 5.6: Different conditions on different effects

Condition	Effect
Number of course groups decreases	• Examination days will be reduced
Balanced number of courses in a course group	• Student sectioning of course decreased • Continuous examinations will be reduced

(i) Number of course groups

Fewer course groups means fewer examination days. Generally, each examination day consists of two timeslots. In order to prevent clashes, each timeslot can only be allocated to one course group. Thus, if the course group decreases in number, the examination days also decrease.

(ii) Number of courses in all groups

When all the course groups have a balanced number of courses, student sectioning and continuous examinations may also decrease. A balanced number of courses means that the difference between the largest and smallest course group is relatively smaller. All the course groups will have a balanced number of courses. If a group has a higher number of courses and most of the courses have a large number of students, then student sectioning is

required when the timeslots do not have an appropriate number of large-sized rooms. For example, a course group has 10 courses and about 5 courses have more than 400 students, then student sectioning is required if the allocated session has only one or two large rooms. Similarly, when there are many students from the same cohort and from different courses distributed into the same group, the possibility of continuous examinations will increase. Meanwhile, if all the course groups have a balanced number of courses, then the same cohort of students can be distributed more widely. Thus, the possibility of continuous examinations can be decreased.

(iii) Number of priority courses

Each dataset is run in both solutions of FESS 1.0 and FESS 2.0. All the datasets will first go through the course grouping stage in both solutions. In both solutions, the courses with a large student number are prioritised and will be grouped first. In FESS 1.0, only the courses with large student numbers have the priority to be grouped first. In the FESS 2.0 solution, the priority concern is added during the course grouping stage. The priority concern is needed for those courses that need to sit for their exams first compared to the other courses. For example, at FEB, all third-year students need to go for their internship after the final examination. Therefore, third year courses are given priority to be scheduled at earlier dates regardless of the size of the course. Using this measure, the third-year students can have more time to prepare for their internship.

Table 5.7: Priority courses at FEB

Dataset	**FEB, 2016/17-2**		**FEB, 2017/2018-1**	
Method	**FESS 1.0**	**FESS 2.0**	**FESS 1.0**	**FESS 2.0**
Priority Courses	• EBD3013 • EBE3093 • EBU1033		• EBA2023 • EBA3053 • EBA3023/EBA3033 • EBF1313/EBM1013	• EBM2063 • EBM3113 • EBQ2024
Total Priority Courses	3		7	
Priority Concern	×	✓	×	✓
Total Exam Days	9	9.5	6.5	7
Day of Last Priority Course	Day 9	Day 3	Day 5	Day 3

As shown in Table 5.7, there are two datasets from FEB with priority courses that need to be considered when generating the final examination timetable. FESS 1.0 does not include the priority consideration, hence courses with priority may finish the examination on the last day of the final examination period. Meanwhile, for FESS 2.0, courses with priority will finish the examination as early as on the third day of the final examination period.

The quality of the result may be affected when there are many priority courses. All the courses with priority will be grouped together instead of being distributed into different groups. Figures 5.10 and Figure 5.11 show the effect of priority courses on the solution quality between FESS 1.0 and FESS 2.0. When the number of priority courses increases, the quality of the solution also increases.

FEB, 2016/2017-2

FESS 1.0

Day	1	2	3	4	5	6	7	8	9	10
AM				EBF3093 EBD3013						
PM									EBU1033	

FESS 2.0

Day	1	2	3	4	5	6	7	8	9	10
AM	EBF3093	EBU1033								
PM	EBF3013									

Figure 5.10: Distribution of priority courses at FEB in Semester 2, 2016/2017

FEB, 2017/2018-1

FESS 1.0

Day	1	2	3	4	5	6	7
AM	EBM3113	EBM2063	EBA3023/ EBA3033 EBQ2024				
PM			EBA3053	EBF1313/ EBM1013	EBA2023		

FESS 2.0

Day	1	2	3	4	5	6	7
AM	EBM2063	EBA3023/ EBA3033	EBA2023 EBQ2024				
PM	EBF1313/ EBM1013	EBM3113	EBA3053				

Figure 5.11: Distribution of priority courses at FEB in Semester 1, 2017/2018

Stage II: Timeslot-Room Allocation

The quality of the timetable is justified by the distribution of courses within the final examination period. The distribution of courses is based on the different programmes offered in the faculty. In FCSIT, there are five programmes offered for each intake. This means that in every semester, there are around 40 student cohorts (5 programmes x 2 intakes x 4 years) to be considered and scheduled for the final examination, whereas the final examination period only consists of 24 timeslots (12 days x 2 timeslots per day).

i) Continuous examinations

A student will have fewer examinations continuously when all the courses for each cohort that are distributed. The analysis on continuous examinations has been plotted in a graph. Figure 5.12 shows the comparison of continuous examinations between the solutions of FESS 1.0 and FESS 2.0 using the dataset of Semester 2 2017/2018. Overall, the consideration of continuous examinations improved the quality of the timetable. With FESS 2.0, the number of two or three continuous examinations were much reduced. As the two or three continuous examinations become fewer, the students of the related courses can have more time to do their revision for the examinations. There is one continuous period of three exams that exists in the timetable generated by FESS 2.0. The three courses are TMT2033 (238 students), TMN2073 (167 students) and TME2133 (99 students). When the course groups are allocated into the available timeslots, the total penalty of all courses in each group is counted to identify the most suitable group that will be allocated. Therefore, the continuous period of three exams cannot be prevented as it has been decreased as much as possible.

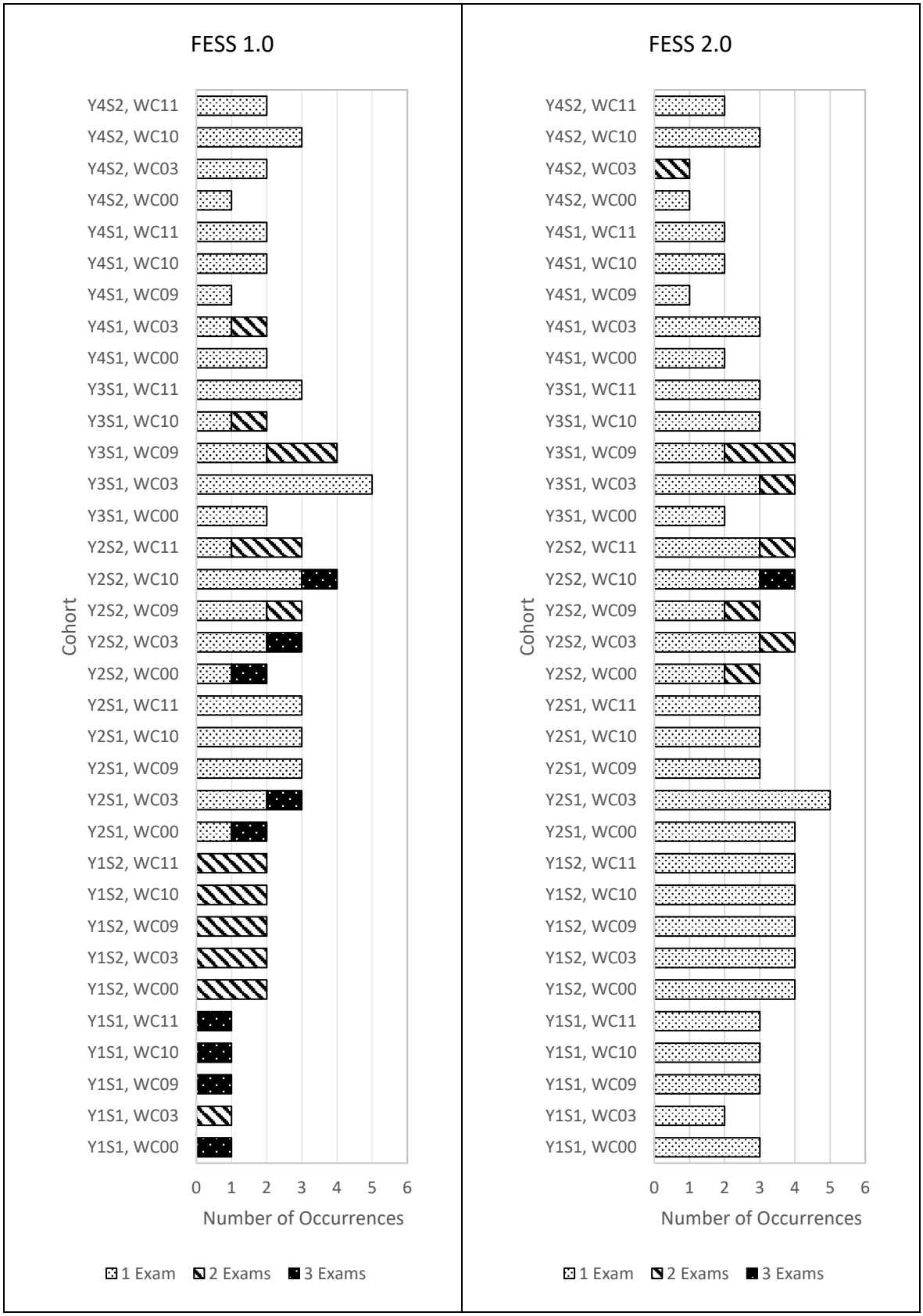


Figure 5.12: Graph on number of occurrences on continuous examinations using FESS 1.0 vs. FESS 2.0

5.5 Sensitivity Analysis

In this section, a sensitivity analysis on the two-stage heuristic algorithm is carried out. The changes in the different factors may affect the quality of the generated final examination timetable. The different factors are course size and course-student enrolment size.

5.5.1 Alternative Solution

In this section, the alternative solution will be tested for the two-stage heuristic algorithm.

Table 5.8: Alternative solution for the dataset of 2017/2018-2 in FCSIT

x	3	4	5	6
Exam Days	6	6	6	6
Total unutilised room capacity	905	925	875	1045
Maximum number of rooms used per course	6	6	6	6

Table 5.8 shows the alternative solution for the dataset of Semester 2 2017/2018 in FCSIT by setting different numbers of checking continuous examinations. With different values of x, the total number of exam days remain at 6 days, total unutilised room capacity around 1000 and the maximum number of room used also remains at six. The different values of x does not affect the quality of the final examination timetable. Similarly, the analysis on the continuous examinations for different values of x is show in Figure 5.13 and Figure 5.14. When x is 3, there is one occurrence of four continuous examinations. When the x is 4, 5 or 6, the maximum number of continuous examinations is three. The different numbers of continuous examinations at different values of x is expected.

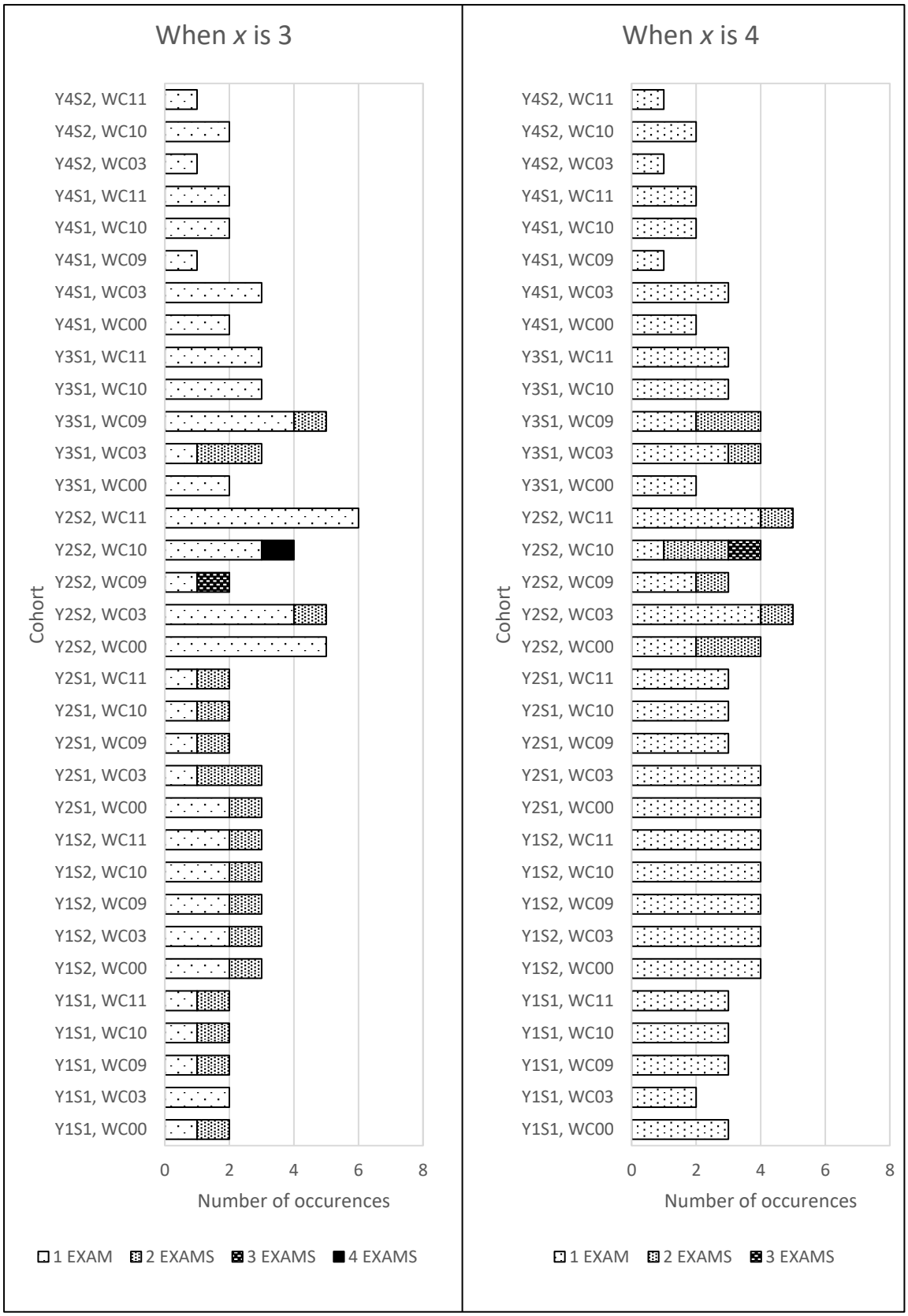


Figure 5.13: Graph for different number of occurrences of continuous examinations

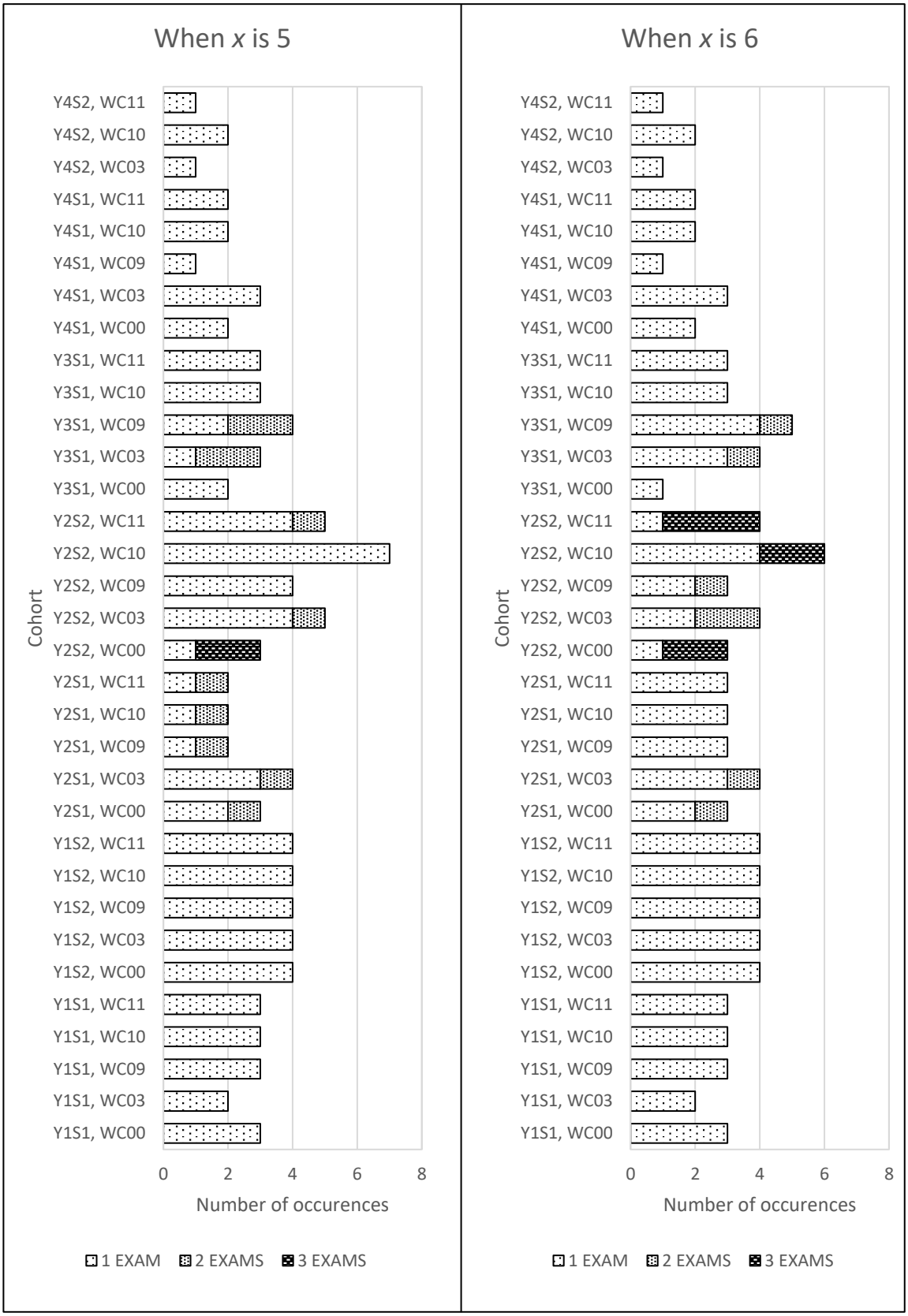


Figure 5.14: Graph for different number of occurrences of continuous examinations

5.5.2 Course Size

In order to check the effect of course size on the solution quality, the courses of two datasets from FCSIT and FEB in Semester 2 2017/2018, were combined. This is to check the effect of an increase in course size and total number of rooms used for the final examination where all the shared rooms used in UNIMAS were added to the new combination of datasets. As shown in Table 5.9, after the combination of two datasets, there are about 129 courses with a total of 11989 course-student enrolment and 22 available timeslots.

Table 5.9: Result of combination of datasets from FCSIT and FEB

	FCSIT	**FEB**	**FCSIT and FEB**
Course	49	76	129
Student	1306	1893	3199
Total Course-Student Enrolment	4403	7586	11989
Timeslot Used	12	16	14
Unallocated Group	0	0	0
Unused Timeslot	10	6	8

The datasets from FCSIT and FEB are two groups of mutually exclusive students. None of the students from FCSIT and FEB are taking the same course. Without the venue restriction, the worst case of timeslots used for the combined datasets must be less than the maximum number of timeslots used among individual datasets of FCSIT and FEB, which is 16 in this case. However, the result obtained was less than the worst-case scenario. In the combined datasets, there were 14 groups of courses which means that 14 timeslots were used for the final examination. The timeslots used were less than the result generated by the individual datasets of FCSIT and FEB. The timeslots used in the combined datasets became fewer when the number of rooms and the number of large room sizes increased. This result shows that the smaller courses from FCSIT and FEB could be combined in a common venue.

Therefore, the simulator can solve different problems with different characteristics of parameters.

5.5.3 Course-Student Enrolment Size

In this section, experiments were conducted on different course-student enrolment sizes. The dataset from FCSIT in Semester 2 2017/2018 was used to check the effect of course-student enrolment size. In the original dataset, there were 49 courses with 4403 course-student enrolments.

Table 5.10: Effect on course-student enrolment size with dataset of FCSIT in Semester 2, 2017/2018

	Original	**Added 100**	**Added 200**	**Added 500**
Timeslot Used	12	12	13	19
Unallocated Group	0	0	0	0
Unused Timeslot	10	10	9	5

Based on Table 5.10, when the course-student enrolment size increases, the timeslots used also increase. When 100 course-student enrolments are randomly added into different courses, the timeslots used remain at 12. However the timeslots used increase when more than 200 students are added to the enrolment. A smaller addition to the course-student enrolment proves that the room is utilised. Meanwhile, it is reasonable if the increment in course-student enrolment leads to an increase in timeslots used. If the increase in course-student enrolment size does not affect the timeslot used, this represents an error in the algorithm. Therefore, the increased timeslots in this situation is a normal phenomenon. Meanwhile, the increase in course-student enrolment size does not affect the unallocated group. As shown in Table 5.10, none of the unallocated groups are in any of the datasets.

This shows that the two-stage algorithm in this research is a robust solution to the timeslot-room allocation.

5.5.4 Room Size

During the timetable scheduling, the quality of the timetable may be affected by the venue resources. Different sets of room sizes were simulated for algorithm evaluation purposes. Generally, there are three different sizes of rooms i.e. ‘big room’, ‘medium room’ and ‘small room’ room. ‘Big room’ is refer to the room has capacity greater than 200. ‘Medium room’ is the room has capacity between 100 and 200. While, ‘small room’ is the room capacity is less than 100. Therefore, the total number of rooms used and the rooms used for each course may be affected by the different room sizes used for the final examination. Therefore, two different datasets were used to test the effect of different room size on the timetable’s quality. The datasets are from FCSIT (Semester 1 2017/2018) and FCSHD (Semester 2 2016/2017). A dataset that adds ‘big room’ (capacity larger than 300) or ‘medium room’ (capacity between 100 and 300) means that whenever any timeslot from the original timetable does not have ‘big room’ or ‘medium room’, then one ‘big room’ or ‘medium room’ will be added to the dataset. Besides that, the dataset from FCSIT was also tested by reducing ‘big room’. However, the dataset from FCSHD was not tested by reducing ‘big room’. This is because the result of room utilisation in the original FCSHD dataset shows that the dataset is not compatible with reducing ‘big room’.

Table 5.11: Effect of room size on dataset of FCSIT in Semester 1 2017/2018

Dataset	**Original**	**Added Big Room**	**Added Big & Medium Room**	**Reduced Big Room**
Total Rooms Used	37	30	30	52
Unutilised Room Capacity	577	1045	945	355

The dataset from FCSIT in Semester 2017/2018 was tested. As shown in Table 5.11, at the original room sizes, 37 rooms were used during the examination. The addition of a room generated a result where only 30 rooms were used for the examination. In contrast, the reduction of a big room in the dataset increased the total rooms used. About 52 rooms were used for the examination when all the big rooms were reduced from the dataset. Similarly, the addition of a big or medium room in FCSHD also had the same effect on the final examination timetable. As shown in Table 5.12, the total rooms used in the room added scenario had also decreased by more than 10 rooms in comparison to the original dataset. Besides that, it is also better in allocating the right venue size for the examinations. When the big and medium rooms were added to the dataset, the unutilised room capacity decreased by about 110. This shows that not only was the total rooms used decreased, but the allocation of the right venue size for the examinations was also better.

Table 5.12: Effect of room size on dataset of FCSHD in Semester 2 2016/2017

Dataset	**Original**	**Increase Big Room**	**Increase Big & Medium Room**
Total Room Used	58	44	45
Unutilised Room Capacity	839	929	729

Indirectly, when the number of rooms used were reduced, the unutilised room capacity also decreased. This is because when the first course in a group was located in a big room, the other courses within that group were also located in the same room as long as the room could fit all the students. However, if the total number of students in a group was

smaller and had a large mismatch with the room capacity, this made the unutilised room capacity higher. Therefore, the unutilised room capacity may not improve along with the improvement in the total number of rooms used.

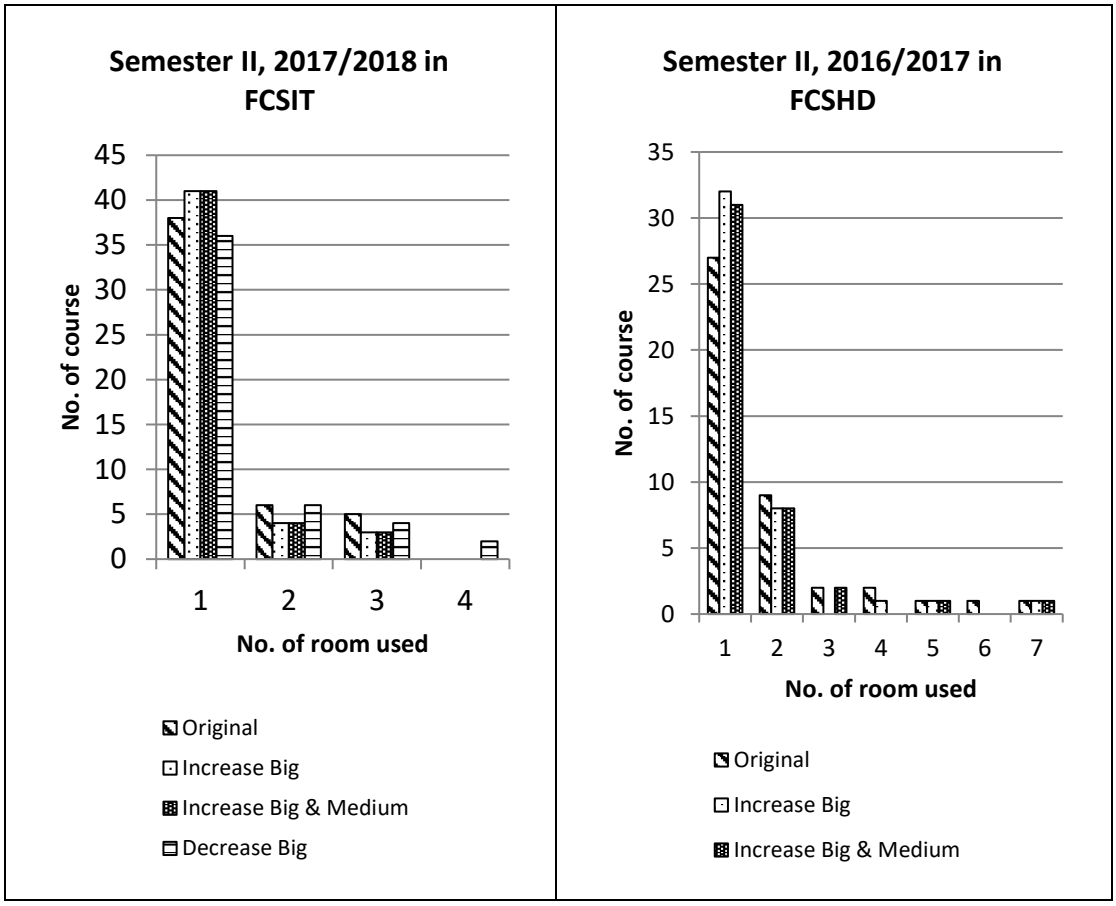


Figure 5.15: Graphs of student sectioning in a course in FCSIT and FCSHD with different room sizes

Figure 5.15 shows the student sectioning in each course with different room sizes in each faculty. As more big rooms were added to the dataset, courses being split in sessions for exams were reduced. However, when the dataset decreased the big rooms, this meant that more courses needed to be split into different rooms. In FCSIT, the addition and subtraction of rooms in the dataset generated a normal result as predicted. However, in FCSHD, the

addition of big rooms did not improve the result as predicted. This is because most of the courses in FCSHD have a large student size. Therefore, the sectioning in a course can be prevented.

5.6 Discussion

The results of the two-stage heuristic algorithm that was applied in the simulation of FESS 1.0 and FESS 2.0 were compared with each other. The algorithm produced a fast solution computing time. Even though more constraints were added in FESS 2.0, the solution was still able to generate a final examination timetable in around 1 minute. FESS 1.0 also had a similar computation time. Besides that, the solution also generated a feasible timetable without clashes. All students will only have one examination at a time.

A few datasets from different faculties in UNIMAS were collected for proposed algorithm validation and performance analysis. Since the venue resources are different among these datasets, sometimes an empty timeslot might exist in FESS 1.0. However, this empty timeslot problem is prevented in the solution of FESS 2.0. The result proved that the two-stage heuristic algorithm in FESS 2.0 produced a practical final examination timetable.

Overall, the student sectioning in FESS 2.0 has improved room utilisation. The maximisation of room utilisation is important to improve the solution quality. For example, if there is one course that uses seven rooms for the examination, it also represents an increase in invigilators on duty. Therefore, the student sectioning in FESS 2.0 decreases the use of many rooms for one course.

In addition, a sensitivity analysis was also conducted in terms of course size, course-student enrolment size and room size. When the course size and course-student enrolment

size increased, the timeslots used also increased. This increment is a normal phenomenon. Similarly, when the room size changed, all the courses were still able to be allocated into suitable rooms. The sensitivity analysis has proven the solution to be robust as changes in course size, course-student enrolment size and room size can still be solved. Furthermore, the alternative solution was also conducted in this section. Different values of x were tested to check the continuous examinations. The analysis of the alternative solution proved that different values of x can still prevent the high number of occurrences of continuous examinations.

CHAPTER 6

CONCLUSION

6.1 Introduction

This chapter concludes the overall implementation of the research project and the contributions of this research on the related body of knowledge. Future work that can be done relating to this research is also discussed.

6.2 Conclusion of Research Study

A study of a university faculty final examination timetabling problem has been carried out in this research. The examination timetabling problem is a NP-hard problem. It is very time consuming and there many constraints that need to be fulfilled when scheduling the final examination timetable.

The objective of this research is to design and develop a computationally bounded multi-objective two-stage heuristic solution with the aim of generating a good quality timetable.

The first stage of algorithm is course grouping. A large number of courses will be grouped into a small number of course groups. Then, the following stage will allocate all the course groups into timeslots and later each course will be assigned into the available rooms. Therefore, the smaller the number of course groups, the shorter the period of the final examination.

The proposed algorithm was then validated and evaluated with real datasets from a number of faculties in UNIMAS i.e. FCSIT, FEB, FSS, FCSHD and FENG. Based on the

results generated from the datasets, the computational results show that the two-stage heuristic solution can generate a final examination timetable in approximately one minute. In terms of room utilisation, the solution managed to improve the room utilisation rate by a 35.09% decrease in the number of rooms used and an 83.99% decrease in unutilised room capacity. Furthermore, most of the *priority course*s can take the examination within the first half of the examination period. For example, at FEB in Semester 1 2017/2018, the examination for the last priority course was held on the third day of the seven examination days. Meanwhile, students continuous examinations was also decreased. In FESS 2.0, there was only one cohort of students that took three exams continuously, whereby in FESS 1.0, nine cohorts of students took three exams continuously.

In the sensitivity analysis, the proposed solution can cope with different sizes of courses, student numbers and rooms. When the course size and student number were increased, the number of timeslots used also increased reasonably. Changes in room size did not affect the allocation strategy. All the courses were still able to be allocated into suitable rooms. All these results show that the proposed solution is effective and robust in solving different sets of data.

6.3 Research Contributions

Upon the completion of this research, there are several contributions that have been achieved:

(a) Developed a fast response multi-objective two-stage heuristic algorithm in solving the final examination scheduling problem. The algorithm not only solved the scheduling problem in terms of computing time and solution quality, but it also saved time for the management team when preparing the final examination timetable.

(b) Developed an automated student sectioning algorithm to utilise the resources and solve the insufficient room capacity. Student sectioning is mainly for courses that have a large number of students and may exceed the room capacity.

(c) Consideration of *priority courses* is included when solving the final examination scheduling problem. All the *priority courses* can finish the examinations as early as possible.

(d) Consideration of continuous exams in solving the final examination timetabling problem. As continuous exams are prevented or decreased, students can have more time to do their revision.

(e) Reduced the number of invigilators on duty during the final examination period. A proper student sectioning and combination of a few courses in the same room can minimise the invigilators on duty.

6.4 Future Work

Currently, the two-stage heuristic algorithm in this research is only up to faculty level, which is local optimum. This is because the available shared venue resources is fixed by university management. Each faculty has a limited use of shared venue resources. Therefore, in future work, this research hopefully can be upgraded to global optimum, which is to study and solve the timetabling problem of an entire university's data. Meanwhile, this represents the whole data of shared room venue resources that will be included into the algorithm. A proper splitting mechanism will make sure of a fair distribution of shared venue resources. Thus, all the shared venue resources can be appropriately used for all examinations.

REFERENCES

Abdul-Rahman, S., Sobri, N. S., Omar, M. F., Benjamin, A. M., & Ramli, R. (2014). Graph coloring heuristics for solving examination timetabling problem at Universiti Utara, Malaysia. *Proceedings of the 3rd International Conference on Quantitative Sciences and Its Applications*, (pp. 491-496).

Aizam, N. A. H., & Uvaraja, V. (2015). Generic model for timetabling problems by integer linear programming approach. *International Journal of Mathematical and Computational Sciences*, 9(12), 718-725.

Asmuni, H., Burke, E. K., Garibaldi, J. M., & McCollumn, B. (2004). Fuzzy multiple heuristic orderings for examination timetabling. *Proceedings of the 5th International Conference on Practice and Theory of Automated Timetabling (PATAT 2004)*, (pp. 334-353), Springer Berlin Heidelberg.

Ayob, M., Malik, A. M. A., Abdullah, S., Hamdan, A. R., Kendall, G., & Qu, R. (2007). Solving a practical examination timetabling problem: a case study. *Proceedings of the International Conference on Computational Science and Its Applications*, (pp. 611-624), Springer Berlin Heidelberg.

Bergmann, L. K., Fischer, K., & Zurheide, S. (2014). A linear mixed-integer model for realistic examination timetabling problems. *Proceedings of the 10th International Conference of the Practice and Theory of Automated Timetabling (PATAT 2014)*, (pp. 82-101).

Boizumault, P., Delon, Y., & Peridy (1996). Constraint logic programming for examination timetabling. *The Journal of Logic Programming*, 26(2), 217-233.

Burke, E. K., Kendall, G., Newall, J. P., Hart, E., Ross, P., & Schulenburg, S. (2003). Hyper-heuristics: an emerging direction in modern search technology. *Handbook of Metaheuristics International Series in Operations Research & Management Science*, 57, 457-474.

Burke, E. K., Eckersley, A. J., McCollumn, B., Petrovic, S., & Qu, R. (2010). Hybrid variable neighbourhood approaches to university exam timetabling. *European Journal of Operational Research*, 206(1), 46-53.

Burke, E. K., & Newall, J. P. (2004). Solving examination timetabling problems through adaption of heuristic orderings. *Annals of Operations Research*, 129, 107-134.

Carter, M. W., Laporte, G., & Lee, S. Y. (1996). Examination timetabling: algorithm strategies and applications. *Journal of the Operational Research Society*, 47, 373-383.

Cataldo, A., Ferrer, J., Miranda, J., Rey, P. A., & Sauré, A. (2017). An integer programming approach to curriculum-based examination timetabling. *Annals Operations Research*, 258(2), 369-393.

Chaudhari, A., & De, K. (2013). Fuzzy integer linear programming mathematical models for examination timetable problem. *ArXiv*, abs/1307.1900.

Chmait, N., & Challita, K. (2013). Using simulated annealing and ant-colony optimization algorithms to solve the scheduling problem. *Computer Science and Information Technology*, 1(3), 208-224.

Cowling, P., Kendall, G., & Soubeiga, E. (2000). A hyperheuristic approach to scheduling a sales summit. *Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling*, (pp. 176-190). Springer Berlin Heidelberg.

Dowsland, K. A., & Thompson, J. M. (2005). Ant colony optimization for the examination scheduling problem. *The Journal of The Operational Research Society*, 56(4), 426-438.

Eley, M. (2006). Ant algorithms for the exam timetabling problem. *Proceedings of the 6th International Conference on Practice and Theory of Automated Timetabling VI (PATAT 2006)*, (pp. 364-382), Springer Berlin Heidelberg.

Hosny, M. (2019). Metaheuristic approaches for solving university timetabling problems: a review and case studies from middle eastern universities. *Proceedings of EMENA-ISTL 2018*, (pp. 10-20), Springer International Publishing.

Jamaluddin, N. F., & Aizam, N. A. H. (2016). Timetabling communities' demands for an effective examination timetabling using integer linear programming. *International Journal of Mathematical and Computational Sciences*, 10(5), 263-268.

Kendall, G., & Hussin, N. M. (2004). A tabu search hyper-heuristic approach to the examination timetabling problem at the MARA University of Technology. *Proceedings of the 5th International Conference on Practice and Theory of Automated Timetabling PATAT 2004*, (pp. 270-293), Springer Berlin Heidelberg.

Mandal, A. K., & Kahar, M. N. M. (2015). Combination of graph heuristic with hill climbing search for solving capacitated examination timetabling problem. *Proceedings of the 4th International Conference on Software Engineering and Computer Systems*, (pp. 118-123). IEEE.

Marie-Sainte, S. L. (2017). A new hybrid particle swarm optimization algorihm for real-world univerisity examination timetabling problem. *Proceedings of the Computing Conference*, (pp. 157-163).

Parker, R. G., & Rardin, R. L. (1988). Discrete optimization. *Academic Press Professional, Inc*, New York.

Pillay, N. (2016). A review of hyper-heuristics for educational timetabling. *Annals of Operations Research*, 239(1), 3-38.

Pillay, N., & Banzhaf, W. (2010). An informed genetic algorithm for the examination timetabling problem. *Applied Soft Computing*, 10(2), 457-467.

Pillay, N., & Ozcan, E. (2017). Automated generation of constructive ordering heuristics for educational timetabling. *Annals of Operations Research*, 275, 1-28.

Sörensen, K., Sevaux, M., & Glover, F. (2017). A history of metaheuristics. *Handbook of Heuristics*, 1-18. Springer.

Wijgers, R., & Hoogeveen, H. (2006). A column generation approach for examination timetabling. *Technical report*. Utrecht: University Utrecht.