

**MODULE REALLOCATION PROBLEM IN THE CONTEXT
OF MULTI-CAMPUS UNIVERSITY COURSE
TIMETABLING**

WANG JIA

NATIONAL UNIVERSITY OF SINGAPORE

2014

**MODULE REALLOCATION PROBLEM IN THE CONTEXT
OF MULTI-CAMPUS UNIVERSITY COURSE
TIMETABLING**

WANG JIA

(M. Mngt., Nanjing Univ.)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF INDUSTRIAL & SYSTEMS ENGINEERING

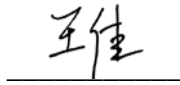
NATIONAL UNIVERSITY OF SINGAPORE

2014

Declaration

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

A handwritten signature in black ink, consisting of the Chinese characters '王佳' (Wang Jia), is written above a horizontal line.

Wang Jia

29 August, 2014

Name : WANG Jia
Student No. : HT080222W
Degree : Doctor of Philosophy
Supervisor(s) : CHEW Ek Peng, LEE Loo Hay
Departments : Department of Industrial & Systems Engineering
Thesis Title : Module Reallocation Problem in the Context of Multi-Campus University Course Timetabling

Abstract

We propose a new type of problems, namely module reallocation problem given timing, which arises from the field of university course timetabling. A new campus is planned and some modules originally allocated on the original campus were to be reallocated to the new campus. Due to practical reasons, the timing was considered as given. The decisions include the module reallocation decision and the room assignment decision. Optimizing the inter-campus traffic is the main objective. We transform stakeholders' requirements into a mathematical model by conducting data analysis on the real data. We propose an iterative two-stage heuristic to solve this problem. This heuristic combines various methods, such as constructive heuristic, clustering analysis, branch and bound framework, Lagrangian relaxation method, etc., to exploit the problem structure and maintain computational efficiency. We also provide a way to fine-tune the timetable to further improve the inter-campus traffic as an extension.

Keywords: University course timetabling, multiple campuses, inter-campus traffic, module reallocation, room assignment, heuristics

Acknowledgements

First of all I would like to express my sincere gratitude to my supervisors A/Professor CHEW Ek Peng and A/Professor LEE Loo Hay for their tremendous help and patience. Their continuous guidance helped me through all the time of the research and the writing of this thesis.

I also wish to thank the Registrar's Office for offering me the precious opportunity to participate in the timetabling project for the University Town of National University of Singapore. In addition, I would like to thank A/Professor NG Kien Ming, Dr. HUNG Hui-Chih, Dr. HE Yaohua, Dr. XIAO Hui, and Dr. WANG Qiang for their collaborations during the aforementioned project.

I am particularly grateful for the assistance given by Dr. LI Haobin.

Last but not the least, I thank my parents for all the endless love and spiritually support they gave me. I also thank FANG Rong and our daughter WANG Suyin for the eternal happiness that they produce.

Table of Contents

Declaration	i
Abstract	i
Acknowledgements	iii
Table of Contents	iv
Summary	vi
List of Tables	ix
List of Figures	x
List of Abbreviations	xi
Chapter 1 Introduction	1
Chapter 2 Literature Review	11
2.1 Overview of Studies on UCTP	11
2.2 Solution Techniques for UCTP	26
2.2.1 General Exact Approaches.....	27
2.2.2 Genetic Algorithm and Other Heuristic Approaches	33
Chapter 3 Data Analysis and Problem Modeling for MRPT	50
3.1 Overview	50
3.2 Data Analysis.....	53
3.3 Problem Modelling.....	65
3.4 Numerical Experiments	78
3.5 Discussion.....	83
Chapter 4 An Iterative Two-Phase Approach to MRPT	86
4.1 Overview	86
4.2 Phase 1: Module Selection Problem (MSP).....	91

4.2.1 Approach 1: Greedy constructive procedure.....	93
4.2.2 Approach 2: Bi-objective MIP model solved by NBI method.....	96
4.2.3 Reparation Mechanism and Local Improvement	102
4.3 Phase 2: Room Assignment Problem.....	104
4.3.1 Overall Framework	106
4.3.2 Dual Bound: Lagrangian Relaxation Method	109
4.3.3 Primal Bound: Constraint Programming-Based Heuristic	116
4.4 Numerical Experiments	118
4.4.1 Numerical experiments related to Phase 1	119
4.4.2 Numerical experiments related to Phase 2	126
4.4.3 Results of the Proposed Heuristic	127
Chapter 5 Fine-tuning on Timing Given the Module Reallocation Decision	134
5.1 Introduction	134
5.2 Methods of time-tuning	136
5.3 Numerical Experiment.....	141
Chapter 6 Conclusion.....	145
Bibliography	153
Appendices	166
A.1 Test Case Generation for Numerical Experiment	166
A.2:Details on the Surrogate Objective Function	176
A.3 Test Case Generation for Numerical Experiment	178

Summary

In this thesis, a new type of problems arising from the university course timetabling is proposed in the presence of the university expansion. The reallocation of modules to the new campus and the incurred impact on inter-campus traffic, rather than the timing, are our main concerns.

In this problem, a new campus is located near the existing campus, and the two campuses are linked by a shuttle bus service. This new campus consists of facilities that are expected to be enjoyed by all students from different disciplines. Optimizing the inter-campus traffic, which measures the level of students' movements for taking courses by travelling from one campus to another, is the main objective. Several considerations for the new campus are addressed by stakeholders, including a good distribution of students for various faculties, a high proportion of the junior students and high resource utilization. The timetable, however, is given by stakeholders who collect corresponding information from individual school/department. Given their timetable, the university would like to know which modules are to be reallocated and which rooms are those modules assigned to. We call this problem Module Reallocation Problem given Timing (MRPT). We modelled MRPT and later solved it by developing an iterative two-stage approach. This approach combines various methods, including constructive heuristic, clustering analysis, branch and bound framework, Lagrangian relaxation method, etc., to exploit the problem structure and maintain the computational

efficiency. We also conducted fine-tuning on the timetable to see whether there is any room for further improvement in the inter-campus traffic.

The main contributions of this thesis contain four parts. First, we propose a new type of problems which arises from the field of university course timetabling. We consider the module reallocation decision and the room assignment decision given a course-timing. The objective function, namely inter-campus traffic, has not been studied before in this area. To understand this objective, we learned from the data that characterize the students' movement behavior. By using similar ways, various requirements from the stakeholders were also finalized and modeled.

Second, from our understanding of the requirements set by the stakeholders, we formulated the problem as a Mixed Integer Programming (MIP) model (The original measurement of inter-campus traffic is non-linear, so we linearized it in the MIP Model). Parameters of objective and constraints were also determined based on the data.

Third, when the problem size becomes large, the commercial solver is unable to solve it. Hence, we propose a heuristics that exploits the good structure of the problem. A decomposition method is used to transform the original problem into a two-stage problem. The first stage determines which modules are allocated to the new campus and the second stage decides which rooms those modules are assigned to.

Fourth, we extend the MRPT by considering that slight modifications to the given timetable are allowed. Based on the selection of reallocated modules, we conduct a local search to find new solutions such that the inter-campus traffic measurement can be improved.

List of Tables

Table 1-1 Examples of different cases that the distance between the old campus and the new campus	4
Table 2-1 Constraints setting in PETP and CBTP	20
Table 2-2 Common select methods in the improvement approach	45
Table 2-3 Common acceptance criteria in the improvement approach	46
Table 2-4 Comparison of commonly used graph heuristics in solving UETP	47
Table 3-1 Number of modules and % of offering faculties grouped by module size range	57
Table 3-2 Distribution of student-module count w.r.t origin of faculties	57
Table 3-3 Distribution of student-module count w.r.t student grade	57
Table 3-4 Five scenarios and their parameter settings	79
Table 3-5 Result summary for 5 scenarios	80
Table 3-6 Result summary for 5 scenarios with other evaluations	81
Table 4-1 Groupings on constraints and decision variables	86
Table 4-2 The count of wins for four scenarios given 1-hour computational budget	121
Table 4-3 The count of wins for four scenarios given 10-hour computational budget	122
Table 4-4 Frequency of approach 1 and 2 being called when solving overall problem	123
Table 4-5 Average Pearson's Correlation Coefficient (PCC) between $F(x)$ and $F'(x)$ with different numbers of modules	125
Table 4-6 The rate comparison in solving phase 2 problem between our heuristic and CPLEX	127
Table 4-7 Results of the count of the success by our heuristic and solver by the grouping of time.	133
Table 6-1 Rules used to generate features of modules	168
Table 6-2 Configuration of modules	170
Table 6-3 The mapping of the non-linear function and its replacement	179

List of Figures

Figure 3-1 Results of sub-clusters in the biggest cluster when $\lambda=50$	64
Figure 4-1 The overall framework of the proposed heuristic	88
Figure 4-2 Obtaining solutions to a typical bi-objective problem using NBI.....	102
Figure 4-3 Flow chart of the branch-and-bound framework in Phase 2	108
Figure 4-4 Plot of the set of $(F(x), F'(x))$ from a 10-module example.....	124
Figure 4-5 % improvement from enabling local improvement	126
Figure 4-6 Example: performance comparison over time on a 800-medium test case	128
Figure 4-7 The traffic measurement comparison.....	130
Figure 4-8 The trend of $N+/N-$ for $ I =400$ case across 24 hrs.....	132
Figure 4-9 The trend of $N+/N-$ for $ I =800$ case across 24 hrs.....	132
Figure 5-1 The detailed inter-campus traffic rate from campus A to campus B	141
Figure 5-2 The detailed inter-campus traffic rate from campus B to campus A	142
Figure 5-3 Changes on traffic rate (A to B) when rescheduling α from FRI to THU.....	143
Figure 5-4 Changes on traffic rate (B to A) when rescheduling α from FRI to THU.....	144

List of Abbreviations

CBTP	curriculum based timetabling problem
CP	constraint programming
GRASP	greedy randomized adaptive search procedure
MRPT	module reallocation problem given timing
MIP	mixed integer programming
MKP	multidimensional knapsack problem
MSP	module selection problem
NBI	normal boundary intersection
PETP	post enrollment timetabling problem
UCTP	university course timetabling problem

Chapter 1 Introduction

The University Course Timetabling Problem (UCTP) is one of the momentous issues faced by the university administrators. Usually, many resources and stakeholders are involved in the timetabling tasks, including students, teachers, rooms and time slots. Typically, many of these resources are subject to various constraints, e.g., limited room capacity, unavailable teachers' timings. At the same time, university course timetabling heavily affects day-to-day campus life for almost everyone in the university. For students and teachers, the timetable greatly determines their study/work schedules every day. For university administrators, a well-balanced timetable which satisfies most requirements improves their management effectiveness. As such, the construction of the university timetable is certainly a very important work.

University course timetabling problem is widely considered as a challenge due to two principal reasons. First, most university timetabling problems are NP-complete problems (Michael and David 1979). Furthermore, this kind of problem is usually of large scale in reality, because a great number of modules are involved in most cases.¹ An automatic timetabling process must be developed to cope with this tedious work. Second, due to the variance among universities, different universities may face different objectives and

¹ In this study, 'module' and 'course' have the same meaning: a series of lessons/classes in a particular subject.

constraints. Even for the same constraint, some universities treat it as a “hard constraint”, while others treat it as a “soft constraint”. Hence, although there are sufficient studies in this field, different works share few common features and may require very different solution techniques when applying to reality.

As far as we know, the first study in this field was by Gotlieb (1962). Since then and during the last fifty years, automatic university course timetabling has been widely studied and numerous papers and articles have been published. In addition, working groups and series of conferences, such as European Association of Operational Research Society on Automated Timetabling and Practice and Theory on Automated Timetabling, have been organized.² Many practical applications have been developed, such as UniTime and WiseTable.³

In recent years, several new requirements have been added in the field of UCTP. The first one is the continuity of the timetable, which means that the revised timetable may not change too much from the previous ones. The main reason for this requirement is that there is no significant demand to produce a brand new timetable as many universities have incorporated the automatic course timetabling systems or systems having similar functions already. Instead, the timetables are revised year after year to meet new needs. If the

² Information can be found via <http://watt.cs.kuleuven.be> and <http://www.patatconference.org>.

³ More information on both solutions can be found via <http://www.unitime.org/> and <http://wisetimetable.com/> respectively.

enrollment is not heavily changed, and if the educational curriculum is stable, the timetables generated from year to year may not change too much. In this case, it is possible that teachers are used to their timings and classrooms.⁴ As a result, if a new timetable is to be designed, a dramatically different one may not be appropriate and could not be accepted by some stakeholders.

The second requirement is to consider the context of multi-campus. Many universities are planning to build new campuses as an important movement in the university development and the response to expanding missions of teaching and researching. Due to various reasons such as visions, accessibility of land, funding, etc., different universities may plan the new campuses in different locations (either remotely or nearby). Table 1-1 shows several examples in terms of the distance between the old campus and the new one. It should be noted that, typically, when the new campus is far away from the old one (a shuttle bus service is usually arranged accordingly), it usually facilitate brand new departments and research agencies. It rarely opens courses for the students from the old campus in a large scale. For this reason, the massive students' movements for taking classes among campuses can be resolved in a trivial way. Unfortunately, these settings are not available in our study.

⁴ We define the compulsory module as a required course for a big group of students according to the curriculum. Hence, the number of takers should be relatively stable from year to year.

Table 1-1 Examples of different cases that the distance between the old campus and the new campus⁵

University	Old Campus and New Distance
Cornell University	Campus in Ithaca and Campus in NYC 1 hour of flight
Binghamton University	Main Campus and University Downtown Center 4 miles
University of Nottingham	University Park Campus and Jubilee Campus 1 miles
Swansea University	Singleton Park Campus and Bay Campus 0.2 miles

New university campus is conventionally designed to host a new department/faculty. However, new ideas about the role of the new campus have been proposed nowadays. One of them states that the new campus should engage all students and researchers together in an integrated way. The main advantage of this new design is that the exchange of knowledge among different disciplines is much easier.

Our study considers a UCTP with aforementioned new requirements based on a university project. In this project, University Town of National University of Singapore has implemented such an idea into practice. It is linked with the main campus, known as Kent Ridge Campus, via a vehicle and

⁵ The information about the four listed new campuses can be found via <http://nyc.cornell.edu/>, <http://www.binghamton.edu/visiting-campus/campus-facilities.html>, <http://www.nottingham.ac.uk/about/visitorinformation/mapsanddirections/mapsanddirections.aspx> and <http://www.swansea.ac.uk/campus-development/>.

pedestrian bridge. An educational complex with residential hostels, teaching facilities and study clusters are provided, which creates an intellectual, cultural and social environment. This design of the new campus promotes an open exchange of ideas and multidisciplinary engagements. Therefore, the new campus is not dedicated to any departments or groups of people. Instead, it is designed to attract and facilitate all students from the main campus to enjoy those wonderful resources. This requirement is very different from the ones discussed in previous studies, because now we need to relocate some of the modules that are previously offered in the main campus to the new campus. On the other hand, National University of Singapore has deployed an automatic timetabling system for several years already. In fact, during our interviews with related personnel, we find that many constraints which are input into the system are decided by individual school/department directly. On one hand, variant curriculums exist at different schools/departments, so there are many intangible and unquantifiable constraints that we cannot easily capture; on the other hand, the schools/departments have built their preferred timetable for flexibility, and these timetables have been used for several years. Any changes to the timing may hugely disrupt this current timetable, and every faculty would be affected.⁶ Many negotiations are expected through the way of finding a solution. Eventually, stakeholders strongly wish that the

⁶ In this thesis, unless mentioned explicitly, “faculty” refers to a group of related departments in a university, e.g., faculty of arts, faculty of engineering, faculty of laws. On the other hand, we use “faculty members” to refer to the lecturers explicitly.

changes on the existing timetable should be as little as possible. The requirements of open-to-all-students and stable timetable bring challenges and are discussed in the study.

The main problem that this thesis studies, namely MRPT, is summarized in the following: Given a timetable, we decide which modules should be reallocated to the new campus and which types of rooms should they be assigned to. At the same time, we want to improve the inter-campus traffic.⁷ In this problem, we study a university which plans the campus expansion. A new campus is built as a new environment attracting students from all schools/departments. A certain number of modules from nearly all schools/departments need to change the venues from the main campus to the facilities on the new campus. Due to practical reasons, the timetable is required to remain the same as given by stakeholders.

In MRPT, our decision mainly considers the objective of optimizing the traffic-flow affected by the students for taking classes for the following reasons: (1) The new campus has an innovative vision as there is no department/schools there. Instead, resources and facilities are shared by all students and faculties. (2) The courses opened on the new campus may attract a large number of students from possibly all schools/departments. They may need to travel across the campus from their last class/for their next class on the main campus. A poorly designed timetable may require too much travelling

⁷ Inter-campus traffic means the traffic related to moving from one campus to another.

time for students and cause the late-for-class. (3) The distance between the two campuses is usually not long, and shuttle buses are commonly used as an important transportation mode. The traffic issue also affects the management of the shuttle bus system so that the overload situation may occur.

On the other hand, our decision is also restricted by the requirements by stakeholders. As the new campus is designed to attract nearly all students, certain way of fairness should be implemented when planning module reallocation. As a result, various constraints are set to fulfill such requirements, e.g., faculty fairness, student preference. In addition, reallocated modules should also ensure a utilization level for facilities on the new campus, especially for those large rooms.

Several challenges are found in this study. We find that the original measurement of inter-campus traffic is non-linear so we need to linearize it. We also learn that the correlation between modules in terms of common students taking both modules is strongly related to the objective value. However, the aforementioned constraints prevent us finding trivial solutions accordingly. In addition, the module reallocation problem can be solved by a commercial solver in small scale. However, when the problem scale gets bigger, the solver cannot handle it. As a result, we propose a decomposition method to transform the original problem into a two-stage problem.

After we solve this module reallocation problem, we extend it by conducting fine-tuning on the timetable to see whether there is any room for further improvement in the inter-campus traffic.

In this thesis, the following contributions are achieved:

- We conduct data analysis to understand the problem better. First we help the stakeholders to find a connection between the students' movement behavior and the inter-campus traffic. Assuming that the students' movement behavior will not be affected by the reallocation of modules, we can use the results from the data analysis to evaluate the inter-campus traffic to make a better decision. In addition, using cluster analysis on these data also provides insights on how to prevent bad solutions, which are later used to build a surrogate measure of traffic. Second, with the understanding of students' enrollment grouped by faculties and academic year, we help the stakeholders to determine the target level of "fairness" for their requirement as well as identify the associated parameters. More importantly, we find that these requirements prevent us from easily generating trivial solutions, such as assigning those courses from the same faculty to the new campus. Hence, it makes the problem more challenging.
- By considering the stakeholders' requirements and analyzing related data, we formulate this real world problem as a Mixed Integer Programming (MIP) model. The original measurement of inter-campus traffic is non-linear, so we need to linearize it. Parameters of objective and constraints are also determined by data mining. As a result, we can develop a model that represents their needs including controlling the traffic while maintaining a set of constraints in terms of fairness.

- We propose a two-stage heuristic approach to solve this problem as it may become intractable when the problem scale becomes bigger. The two stages, namely the module selection stage and room assignment stage, are derived by exploiting the problem structures. In the first stage, we introduce a multi-objective framework to tackle the problem, as the selection of modules is affected by not only the traffic but also stakeholders' requirements. Under the multi-objective framework, we propose two methods to generate a solution. The first heuristic is a greedy constructive method based on balancing between the objective value and the violations of constraints. The second heuristic constructs a bi-objective model, which uses a surrogate measure of traffic based on clustering analysis on the student-module registration data. This model is solved by the Normal Boundary Intersection (NBI) method. In the second stage, we use a branch and bound framework to solve the problem. Within this framework, we use the Lagrangian relaxation method to solve the sub-problem, in which we can identify a knapsack-type structure and thus the sub-problem can be solved efficiently. We also use constraint programming techniques to help find the incumbent solution.
- We further extend the module reallocation problem by considering that the timetable is allowed to be modified slightly from the given one. By keeping the selection of reallocated modules unchanged, we conduct a

local search to find new solutions such that the inter-campus traffic measurement can be improved.

In the subsequent part of this thesis, we firstly review the studies in the mostly-related field, i.e., UCTP in Chapter 2. The key elements, concepts, and various solution techniques related to UCTP are summarized. In Chapter 3, we describe the data analysis works to understand the stakeholders' requirements in MRPT followed by our mathematical model. In Chapter 4, we propose an iterative two-phase approach to solve MRPT and demonstrate the related numerical experiments to compare the performance between our proposed approach and the commercial solver. In Chapter 5, we extend the problem by conducting a fine-tuning process on the timetable given the solution to MRPT. In Chapter 6, we address the conclusion of this study as well as providing possible directions for future studies.

Chapter 2 Literature Review

As we believe that our problem is still in the context of university course timetabling problem, we focus our review on studies of UCTP, and discuss on several interesting topics which show connections to our study. In this chapter, we give an overview on the scope of UCTP and various objectives/constraints from different problem instances in previous studies. This overview helps to understand the mathematical challenge of this problem and the importance of capturing the correct requirements in modelling UCTP-type problems. We then discuss the solution techniques for UCTP by grouping them into the exact approaches and the heuristic approaches.

Although our study has not been completely tackled in previous studies, ideas from literatures still provide insights. Some hard constraints, soft constraints and problem modelling ideas are related to our study in Chapter 3. Topics such as decomposition from the timetabling problem to room assignment and timing problem, branch-and-bound framework and genetic algorithm applied to timetabling problem are also related to our study in Chapter 4. The concept of conflict in timing is closely related to our study in Chapter 5.

2.1 Overview of Studies on UCTP

In this section, we present an overview of the university course timetabling problem. We explain the key elements in the problem. We list the common constraints discussed in previous literatures. We then describe two important

tracks in recent years and show the difference in terms of objectives and constraints. Both tracks share an important topic, namely graph coloring, which is one of the most important sub-problems. For this reason, we then show its definition and various algorithms to solve this problem. With key elements of UCTP described, we briefly introduce a recent trend of modeling UCTP as a multi-objective problem.

A UCTP is defined as a problem which assigns \mathbb{C} , a set of events of courses, into T and R , which are a set of timeslots and a set of resources respectively. A solution $x \in X = \{c_i, t_i, r_i : c_i \in \mathbb{C}, t_i \in T, r_i \in R\}$ has to satisfy a set of constraints, i.e., $x \in \mathbb{Q}, \forall x \in X$. The constraint set \mathbb{Q} can be even categorized into hard constraints \mathbb{Q}^H and soft constraints \mathbb{Q}^S . Hard constraints are those requirements that a solution must satisfy, while soft constraints serve the similar role as objective functions in an optimization problem. In the following, we describe the three elements: \mathbb{C} , T , R .

Key elements

An event⁸ is the session of some course taken place in one room and typically in one or two timeslot. An event may have the following features: a name to identify itself from other sessions if applicable, a piece of information about the takers (e.g., the size of the event, the department/school offering this event), and the lecturers. One course may have more than one event, including

⁸ In this thesis, *event* and *class* has the same meaning as a period of time during which someone teaches a group of people.

lectures, tutorials or laboratories. Events belonging to the same course may be connected by some requirements in the UCTP. For instance, the lecture sessions are preferred to be taken place in the same room across a week.

A timeslot is a period to contain one event. Unlike the concept in the scheduling problem, time in UCTP is discrete and grouped into slots. Common timeslot could be 1-hour slot, 50-minute slot, 45-minute slot, etc. Moreover, timeslots recur from week to week in UCTP. The number of timeslots per week is limited (e.g., 45 one-hour-long slots per week providing nine working hours in 5 working days) and normally indexed in chronological order. In an individual timeslot, at most $|R|$ events are assignable. Note that there are many real-world cases when each event requires different length of timeslots. For instance, one event requires two consecutive⁹ timeslots. In this case, one common way (See, e.g., Schaerf 1999; Lewis 2008; Qu et al. 2009; MirHassani and Habibi 2011) to handle is to split the long events into two shorter events and enforce a requirement on consecutiveness.

A room is a resource that facilitates an event. Rooms have different sizes and possibly different purposes (i.e., only a subset of \mathbb{C} is eligible to be assigned to a specific room). In the literature, rooms are usually individually indexed even when some of them are similar or identical. In an individual room, at most $|T|$ events are assignable.

⁹To clarify, the two timeslots are consecutive only when they are on the same day.

In some circumstances, lecturer is also considered as an element. However, note that in many cases of UCTP a lecturer has already been assigned into the course(s). Therefore, constraints related to lecturers can then be converted into the other forms (for instance, lecturers' preferred timeslots and/or preferred rooms). This essentially defines some eligible set of "feasible" assignments of the three key elements. As a result, in this thesis lecturer issue is not explicitly described.

Constraints commonly considered

Obviously, $|C| \leq |T| \times |R|$ is a necessary condition for the existence of a timetable solution. However, this inequality is very loose comparing to the constraint set \mathcal{Q} . In previous literature on UCTP, a large number of constraints have been addressed arising from different problem instances. At this stage, we only summarize those commonly cited ones:

Hard constraints

HC1: (*time conflict constraint*) Two events should not be held in one timeslot once there are common takers, i.e., no student should attend two events at the same time. It is sometimes referred to as stable set constraint (White and Pak-Wah 1979, Tripathy 1984).

HC2: (*room occupancy constraint*) A room cannot hold more than one event in one timeslot (Carter, Laporte, and Chinneck 1994, De Causmaecker, Demeester, and Berghe 2009).

HC3: (*room capacity constraint*) Events should be assigned to rooms of sufficient sizes. If this constraint is considered as a soft constraint, it means the number of students left without a seat for all the events is to be minimized (Di Gaspero and Schaerf 2001, Burke, Marecek, et al. 2010).

HC4: (*room compatibility constraint*) Events should be assigned to rooms providing appropriate features. For instance, a class which requires special equipment should be assigned to those rooms that are able to provide it (Ceschia, Di Gaspero, and Schaerf 2011).

HC5: (*time availability constraint*) Events should be assigned to timeslots that are available. The availability may mostly depend on the corresponding lecture's availability (Stallaert 1997).

HC6: (*time precedence constraint*) Events should be allocated according to the event precedence relationship, i.e., one event should be scheduled earlier than the other (Drexl and Salewski 1997).

HC7: (*event completeness constraint*) Every event should be assigned into a room and a timeslot. Notice that this constraint is tread as a soft constraint in some circumstances (Lewis, Paechter, and McCollum 2007).

Soft constraints

SC1: (*late event constraint*) Students should not be assigned to the last timeslot of the day, i.e., those last timeslots of a day should be scheduled with the lowest priority (Ceschia, Di Gaspero, and Schaerf 2011).

SC2: (*dispersed event constraint*) The consecutive events that a student attends may not exceed a specific number, normally 2 (Perzina 2007).

SC3: (*isolated event constraint*) The case that a student only takes one module in a day should be prevented (Schaerf and Di Gaspero 2007).

SC4: (*inter-site travel constraint*) As a large university may be split into several campuses, the occasion of that two events, which are held on different campuses, are scheduled consecutively shall be avoided. This constraint was first proposed in studies by Lewis, Paechter, and McCollum (2007) in the discussion section. However, as far as we know there are no other related literatures studying on this specific constraint.

SC5: (*minimum lecture working days*) The timeslots assigned to all lecture sessions of one course should be spread into a number of working days specified by a lower bound, e.g., three days. In other words, the overall number of the positive difference between the number of actual days assigned for each course and the corresponding bound should be minimized (Burke, Kendall, and Soubeiga 2003; Daskalaki, Birbas, and Housos 2004).

SC6: (*curricular compactness*) If two events belong to the same curriculum, they should be assigned to consecutive timeslots. In other words, the sum of all occurrences of isolated events which belong to the same curricular should be minimized (White and Zhang 1998; Di Gaspero, McCollum, and Schaerf 2007a).

SC7: (*lecture room stability*) For all the lecture sessions of one course, the allocated room should be the same. In other words, the number of distinct course-room allocation minus the number of courses should be minimized (Burke, Marecek, et al. 2010).

All these hard and soft constraints mentioned have no direct connection with either the objective or the main constraints proposed in our study. However, some studies share connections with our study to some extent. For instance, when enforcing SC4, it may help resolve the inter-campus traffic. In this case, the back-to-back modules that involve same students are preferred to be allocated into the same campus. However, this soft constraint is a special case that has been considered by our proposed objective function. Apart from back-to-back modules, our objective function also considers those module pairs which have more in-between time and may also contribute significantly (for instance, two highly-related modules which have more than 100 students in common and the in-between time is roughly 1 hour) to the traffic. These module pairs cannot be considered by simply enforcing SC4.

Two tracks in UCTP

There are two important tracks in UCTP field, namely Post Enrollment Timetabling Problem¹⁰ (PETP) and Curriculum Based Timetabling Problem (CBTP). The grouping into two tracks was first introduced in International Timetabling Competition. It should be noted that in many real world situations, the construction of a departmental/institutional course timetable involve a combination of curricular-based and post-enrolment features, as well as iterative negotiations with teaching and administrative staff.

In PETP, the timetable is produced after student enrolment on courses is over, so the space for error is little. Maximum student satisfaction and good utilization of resources are to be achieved.

In CBTP, the weekly timetable of the lectures for various courses within a given number of rooms and periods is generated, where conflicts between courses are defined according to the curricula.¹¹ Therefore, lectures in the same curricular must be allocated into different timeslots.

It should be noted that the major difference between PETP and CBTP does not come from the process of collecting ‘conflict’ data between two

¹⁰ The definition of PETP can be found from studies by Lewis, Paechter, and McCollum (2007) and http://www.cs.qub.ac.uk/itc2007/postenrolcourse/course_post_index.htm, while the definition of CBTP were given by Di Gaspero, McCollum, and Schaerf (2007b).

¹¹ Note that the definition of curriculum in the CBTP (denoted here as curriculum¹) is different from the ordinary definition. Generally, a curriculum refers to a set of courses a student needs to take in order to get a degree throughout his study. However, the concept of curriculum in CBTP requires that the student following the same curriculum all take the same courses in any time, while in general students may be given more flexibility to choose the order of taking courses.

tracks. In fact, the conflict matrix can be obtained with no significant difference in both cases. The major differences, however, come from two other factors. First, PETP deals with individual lecture, as the conflict between lectures can be obtained from enrolment data. CBTP, however, deals the conflict from the grouping of courses.¹² In fact, a course in CBTP is possibly composed by multiple lectures (each is taken by the same group of students and all should be assigned to different timeslots). Second, the hard constraints and soft constraints are defined very differently. In Table 2-1, we summarize typical hard constraints and soft constraints with their presences in PETP and CBTP.

¹² It should be noted that some studies even state that in PETP the confliction is more severe than in CBTP (Burke et al. 2012).

Table 2-1 Constraints setting in PETP and CBTP

Constraints	PETP	CBTP
HC1: time conflict constraint	HC	HC
HC2: room occupancy constraint	HC	HC
HC3: room capacity constraint	HC	SC
HC4: room compatibility constraint	HC	
HC5: time availability constraint	HC	HC
HC6: time precedence constraint	HC	
HC7: event completeness constraint		HC
SC1: late event constraint	SC	
SC2: dispersed event constraint	SC	
SC3: isolated event constraint	SC	
SC4: inter-site travel constraint	* ¹³	
SC5: minimum lecture working days		SC
SC6: curricular compactness		SC
SC7: lecture room stability		SC

As a comment, we discuss the conflict-information collection process of these two tracks. For PETP, it comes from the students' choices collected before the timetabling and CBTP from the curriculum. The conflicting information involves different levels of students' choices for both problems: The one from the curriculum can be viewed as choices of compulsory courses for the students associated, and CBTP should capture most of it, as the degree requirements should be stable from year to year. In addition, students also want to choose selective courses in the university, which may not affect their acquiring the degree, but could enrich their knowledge and experiences.

¹³ These constraints have not been considered yet by Lewis, Paechter, and McCollum (2007) but is highlighted to attract attention in the future study.

Confliction may also arise from these selective courses and the “post-enrollment” process may be able to capture them. However, the way that students make their choices on selective courses should be flexible: We believe that, apart from the pure preferences on the course itself, many other factors may play important roles as well. The timing of courses, for example, should be one of them, as the students may wish to choose those selective courses without violating their week plans. Nevertheless, since the timing is not given when the students provide the choice-of-the-course information, the corresponding choices of selective courses in PETP may not precisely reflect the students’ real choices, and the later generated timetables may not satisfy students due to the data inaccuracy. In fact, it could be expected that some students may simply choose all the preferred courses, ignoring the timing preference, to increase the potential satisfactory. As a result, the conflict could become worse, and the overall timetable becomes harder to plan. In addition, the difference between the two collection processes also explains why some constraints are considered in one track but are not in the other. For instance, as CBTP mainly considers lectures, the incompatible module issue should be naturally resolved as most rooms in the university can cater lectures. Another example is that HC3 is considered as soft constraint in CBTP. We believe that the main reason is that if the constraint is violated, it can be resolved by splitting the module (remind that in CBTP a module may involve a series of lectures) and rearranging the rooms if possible (additional room may be used).

Graph Coloring

To be clear, graph coloring is not closely related to our study as the timing is pre-fixed and the conflict issue does not exist in our study. However, this topic is so important in the field UCTP that we have to summarize it in the following.

To group the events into timeslots without any conflict is a (vertex) graph coloring problem. Since the property of timing-conflict-free is considered as one of the most vital requirements a timetable solution must have, graph coloring is one of the key topics in UCTP.

In the derived graph, a vertex represents the event, and an edge exists when two events are in conflict. A “valid” coloring is an assignment of vertexes with colors such that every two adjacent vertexes are colored differently. The graph coloring problem is defined to find the coloring with the least number of colors. One can define other types of graph coloring problem such as k -coloring which at most k colors can be used. In this case, weights could be set as the number of students in conflict. The objective function is minimizing the accumulated weights of violated edges. Solving graph coloring using exact approach currently needs exponential time (Byskov 2004). In practice, greedy coloring is usually used to speed up the computation. A greedy algorithm considers the vertexes of the graph one by one and assigns each vertex a first-fit color. The sequence usually reflects the “difficulties” of vertex coloring. This idea has been used to develop various graph-based heuristics in timetabling field.

Note that graph coloring is only a sub-problem of the complete timetabling problem, as the solution merely “groups” the events. Besides the room assignment, the timeslot assignment additionally requires a mapping from colors to the timeslots. In other circumstances, extra constraints are addressed, and it makes the complete problem more difficult.

Multi-objective UCTP

In many practical problems, more than one objective is involved, and these objectives are usually conflicting with each other. This issue exists in UCTP as well. More than one soft constraint tends to be considered in recent studies. For instance, the violation of SC5 and SC6 cannot be resolved simultaneously since the former constraint requires a dispersed time assignment of lecture sessions of the course, while the latter constraint requires a compact one. In our study, we also face a multi-objective optimization sub-problem which is addressed as the module selection problem in Chapter 4. Therefore we summarize the common solution techniques for solving multi-objective problems in the following.

The first group of methods is transforming the multi-objective problem into a single-objective problem by using scalarization (Ismayilova, Sağır, and Gasimov 2007; Geiger 2009). A well-known example is weighted sum approach, in which the weights are usually positive. By varying different weights, this approach performs well for problems having convex objective space. A critical issue is that this approach may lead to a single solution that may not be particularly useful when decision makers want to examine

tradeoffs of different objectives among multiple solutions. In addition, when the objective space is not convex, scalarization method may not explore some regions of pareto frontier and the “sampling” by using different weights/scales is not generally evenly distributed (Hwang and Masud 1979).

The second group of methods utilizes the “preference” information on individual criterion. Specifically, methods which can be categorized in this group requires some “rank” information which means one is strictly more important than the other. Typical methods include utility function method, goal programming and lexicographic method (Ulungu and Teghem 1994).

The third group of methods uses modified meta-heuristics which are able to consider multiple criteria. We use two examples to show how meta-heuristics can be adopted. The first example is Generic Algorithm (GA). GA is a popular meta-heuristic approach in solving UCTP and it has the ability of generating multiple solutions simultaneously. According to the survey by Konak, Coit, and Smith (2006), the main change to traditional GA is on the fitness computation, i.e., how to select the solution into the parent population.¹⁴ A Pareto-based ranking scheme on population selection stage is

¹⁴ In most methods in the multi-objective GA context, the ranking, fitness assignment and selection are applied before the GA operations. With the parent population generated, the GA operators such as crossover and mutation generate the offspring population, which has enough number of solutions for the population of the next generation. Usually no extra selection procedure is applied to this offspring population.

commonly used, such as Non-dominated Sorting Genetic Algorithm-II (NSGA-II) (Deb et al. 2002) and Strength Pareto Evolutionary Algorithm 2 (SPEA2) (Zitzler et al. 2001). In addition, a good selection design has to maintain both the diversity and the elitism in the population (Carrasco and Pato 2001). The weakness of this approach is that the computational speed is too low. On the contrary, modified Greedy Randomized Adaptive Search Procedure (GRASP), proposed by Martí et al. (2011), can achieve higher computational speed. In this construction method, various criteria are selected in each construction step by given chances. Since the GRASP is widely acknowledged for its simplicity, efficiency and the ability to escape from local area, the modified GRASP is also light-weighted and fast to generate a solution.

The forth group of methods tries to explore the Pareto optimal solutions by using mathematical programming. These methods have the ability to generate (weak, in many cases) pareto-optimal solutions, but typically consume more time comparing to the first three groups. Several methods have been proposed, including Normal Boundary Intersection (NBI) (Das and Dennis 1996), Modified Normal Boundary Intersection (Shukla 2007) and Normal Constraint (NC) (Messac, Ismail-Yahaya, and Mattson 2003). Specifically, modified NBI improves NBI in proving pareto-optimality and was reported to achieve better computational efficiency when comparing with NC (Motta, Afonso, and Lyra 2012).

In our study, we try to solve a multi-objective sub-problem in two different approaches and use an intelligent selection method to call each approach adaptively. The first approach is a simple and quick method, while the second one is complex but can generate better solutions. As for the first approach, we use the modified GRASP due to its simplicity and high efficiency. We do not choose modified GA due to its low computational time. Also, we do not choose scalarization method because the objective space in our study is typically discrete. In addition, in our case the stakeholders literally give the same preference to each constraint (and some of them are considered as objectives in Chapter 4), the second group of approach is not adopted in our study either. For the second approach, we use modified NBI for the reasons highlighted above.

2.2 Solution Techniques for UCTP

We categorize the solution techniques for UCTP into two groups: One is exact approaches, and the other is heuristics. In the first group, UCTP is treated as a MIP problem, so the solution techniques naturally include those based on branch-and-bound methods and various decomposition approaches. In the second group, we discuss the genetic algorithm, which is primarily used as a single meta-heuristic in this field. We then discuss hyper-heuristics, which intelligently combines various heuristics together.

2.2.1 General Exact Approaches

Exact approaches to UCTP mainly refer to the ways that build and solve the mathematical model. There are various mathematical models built for many problem instances described in the previous section, and most of them fall into the range of Integer Programming (IP). Branch-and-bound framework is dominantly used. However, due to the large number of variables/constraints in UCTP, the branching tree cannot be fully explored due to the big problem size. Therefore, branch-and-bound is usually combined with other techniques.

In general, the key to the good performance in using branch-and-bound-based approach is a “good” formulation¹⁵ of the problem, i.e. the formulation is preferred to be close to the convex hull of the feasible set so that the duality gap is decreased, and the fewer rounds of branching is needed. The cost of generating a good formulation is that much more constraints are needed to approximate the convex hull (Wolsey 1998). Nevertheless, by following such an idea, there are generally two groups of approaches in related studies: (1) Outer approximation approach approximates the convex hull by intersecting half-space. One example is cutting plane method. It generates valid inequalities determined by a separation algorithm to improve the formulation. Branch-and-cut method, which integrates cutting plane method into branch-and-bound, generates tighter dual bounds at the node in the

¹⁵ A formulation for a IP $\max\{cx, x \in X, X \subset Z^n \times R^p\}$, a polyhedron $P \subseteq R^{n+p}$, which is described by a finite set of linear constraints, is a formulation for the set X in the IP iff $X = P \cap (Z^n \times R^p)$

branching tree. (2) Inner approximation approach approximates the convex hull by supplementing partial description. Examples include column generation and Lagrangian method. Column generation dynamically introduces new columns, which is determined by solving a pricing problem. Branch-and-price, which integrates column generation into branch-and-bound, is used for those problems with huge amount of variables. Lagrangian method, on the other hand, mainly deals with structured IP with complicating constraints and transfers the problem into a series of Lagrangian relaxation problems with dual parameter. It solves the duals using a sub-gradient search rather than solving the restricted master problem in column generation.¹⁶ In addition, it can also be integrated into a branch-and-bound framework by developing primal heuristic, and this idea is adopted in our study. For more detailed technical description, see the literature by Galati (2010).

In the studies on UCTP, the aforementioned approaches have been frequently applied in recent studies. The main motivation is that the natural/monolithic formulation, even for those problems with only a few set of hard/soft constraints, has been reported to be weak and requires excessive iterations to find optimal.¹⁷ For those studies using cutting plane method, various parts of “natural” formulation have been reformulated. In the

¹⁶ Since no efforts to solve a primal sub-problem and no primal solution information explored, Lagrangian method is usually quicker than column generation.

¹⁷ A formulation merely represents the problem requirements and in most cases also a “compact” formulation with a polynomial number of variables and constraints

following two paragraphs, we give details on studies using branch-and-cut and branch-and-price in the following.

For branch-and-cut approach in UCTP, except for HC4-HC6, nearly all other hard constraints can be used to imply better cuts or bounds. For instance, HC1 essentially describes a conflicting graph, and the properties in the graph could be used to develop cuts known as cuts from graph coloring (Campêlo, Corrêa, and Frota 2004). Among them, clique inequality has been reported to best strengthen the formulation. This inequality ensures that every event in a clique must be assigned with different timeslots. Another example is the lifted odd-hole cut (Avella and Vasil'Ev 2005). In addition, various cuts can be derived from soft-constraints. For instance, “natural” formulation of SC7 (See constraint (14-15) in Lach and Lübbecke 2012) has been replaced with a new set of constraints with less decision auxiliary variables and enumerated possible patterns. Hence, it becomes a strengthened formulation (See constraint (16) in Burke et al. 2012).

For branch-and-price approach in UCTP, comparably fewer studies have adopted this method. An early study by Papoutsis, Valouxis, and Housos (2003) were on a school timetabling problem. Later, a study by Qualizza and Serafini (2005) was on solving UCTP.¹⁸ In these studies, a column was defined as the course-timeslot-room pattern, which is an assignment of all the

¹⁸ However many hard/soft constraints are not considered such as HC3 and most of the soft constraints. The objective is merely overall preference on the assignments.

required hours for the course to definite timeslots and definite classroom types. Common approach of branch-and-price procedure is used.

In our study, we use the method of branch and bound as the general framework to solve an IP problem in Chapter 4, and combine the Lagrangian method by using it to obtain the dual bound. The main reason that we use this combination is that we can exploit a good structure once we relax certain constraints. In addition, as the numbers of rows and columns in that IP problem are generally similar, the beneficial of using either branch and bound and branch and price is questionable.

Decompositions Techniques

Even by employing various techniques, exact approaches may still be unable to tackle many problem instances of UCTP. In general three main difficulties have been reported. First, the IP solver is unable to handle the very large scale (e.g., hundreds to thousands of events, thousands of students and lecturers) and complex constraints (e.g., a large number of different hard constraints/soft constraints) in practice (Murray and Müller 2007). Second, the performance is not robust considering that the requirements in reality may change from time to time (Burke, Marecek, et al. 2010). Third, a good feasible solution is sometimes too difficult to obtain, even after tweaking the parameters of the solver, e.g., node/variable selection strategy, call frequency of a primal heuristic at node (Burke, Marecek, et al. 2010).

As a result, decomposition is needed to decrease the dimension of the problem by solving a series of sub-problems so that the solver can handle. In fact, several problem instances of UCTP often offer a good structure for decomposition. For the objective functions, they are usually in the form of the weighted sum of penalties associated to the soft constraints. With carefully defined decision variables, the objective is separable.¹⁹ To demonstrate this idea, we show two decomposition directions in CBTP. The first one ignores the room assignment decision first and considers it later, while the second one generates some room assignments first and later determines the valid timing assignment.

In Burke, Marecek, et al. 2010, the original problem was decomposed into two parts. The former part was called “surface”, and the latter part was called “dive”. In surface stage, a relaxed-problem was considered, in which HC2 and SC7 were ignored. The room assignment decision was hence not considered.²⁰ However, to guarantee a time assignment yield a feasible room assignment, a validity constraint was added. The surface problem was solved by solver much quicker. Then with those solutions to the surface problem sorted by objective value increasingly, a number of divers were applied on to each. The diver was constructed as IP with different “solution” constraints (in

¹⁹ Specifically, different soft constraints depend on different decisions: S1-S3, S5-S6 rely only on timing assignment, S7 only rely on room assignment.

²⁰ The author suggests that one can enumerate all possible scheme of ignoring terms in objectives by doing some numerical experiments beforehand to get prior information. An automatic approach of generating surface could be a future direction.

terms of the solution to the surface). Once a better dive problem was solved, the upper bound was updated to cut off later diver instances. The final solution was, of course, not a global optimal, but not that the lower bound was also provided. This approach is different from those heuristic approaches which iteratively improve the solution, but from numerical experiments the performance is very promising.

On the other hand, the only constraint related to room allocation of lectures is HC2. Moreover, it is possible to determine a time allocation of lectures first (the first stage problem) in such a way that the feasible room allocation satisfying HC2 can be always generated by enforcing constraints. After that, one only needs to determine the perfect match from each lecture to a room for every timeslot (the second stage problem). The constraints enforced to implicitly satisfying HC2 during the first stage problem is related to the Hall's theorem which restricts the number of lectures can be assigned to each timeslot according to the given lecture-room fitting relationships, e.g., lectures may only fit a subset of rooms (Jiang and Nipkow 2013). The advantage of this decomposition is that the number of constraints derived by Hall's theorem in reality can be restricted to a reasonably scale. Therefore, the first stage problem as a IP is not hard to solve (Lach and Lübbecke 2008), and the second stage problem (a series of perfect matching problems) is very easy to solve in polynomial time without considering SC7. Notice that SC7 encourages that the same room is preferred to be assigned to multiple lectures belonging to one course. As a result, the solutions to the perfect match problem in the second

stage problem may not be the global optimal one with the new soft constraint considered. With respect to the solving methodologies for the second stage problem, one now has to come back to resort to a general IP model rather than a simple LP.

In our study, none of the aforementioned decomposition techniques are directly used. This is because our study has a different set of constraints and the timing is fixed. However, in an indirect way, the idea of decomposition in our study shares some similarity with the one of relaxing HC2 in previous studies. In both ways, constraints which consider a different set of decision variables are identified and the decomposition is performed on them. In our studies, we identify a set of constraints which only considers room allocation while the others consider module reallocation decisions. By relaxing those constraints first we are able to determine module reallocations first. We then determine the room allocation by reconsidering these constraints.

2.2.2 Genetic Algorithm and Other Heuristic Approaches

When the problem scale gets bigger, the effectiveness of the exact approach may deteriorate fast. On the other hand, obtaining the true optimal is not very important. Instead, a good feasible solution may already meet the requirement of stakeholders, and sometimes stakeholders prefer to be presented with more than one option. In these cases, heuristic approach becomes more appropriate to solve UCTP. In this section, we first discuss Genetic Algorithm (GA), which is chosen due to its popularity among variant heuristics in this field. We demonstrate how GA can be adopted to solve traditional UCTP and multi-

objective UCTP. We then discuss the hyper-heuristics that may overcome the weakness from individual heuristic by incorporating and intelligently selecting several simple heuristics. Although these heuristics cannot be directly applied to our study, the ideas such as the constructive approach and local reparation share similarity with our research.

GA for UCTP

GA is a population-based search algorithm that typically concentrates more on exploration than on exploitation. GA has been successful applied in solving UCTP from many literatures. We describe two key topics to show how GA is adopted to solve UCTP. We first discuss the chromosome encoding method, because a chromosome encoding which captures the structure of UCTP may help increase the performance of GA greatly. We then discuss the way that GA is combined with neighborhood-search techniques in order to improve the search performance.

Chromosome encoding is vital to finding a good solution using GA. To solve timetabling problem, there are three chromosome representations commonly used in the literatures, namely traditional representation²¹, permutation representation and grouping GA representation.

For traditional representation, chromosome usually stores the information of the assignment for each event explicitly, and ordinary GA

²¹ This representation is also called *literal encoding* by Davis (1991), and straightforward encoding by Falkenauer (1997).

reproduction operators²² are used, such as one-point/two-point crossover. The encoding usually uses an array. The array can be single-dimensional or multi-dimensional. Take the example of a one-dimensional array, the element refers to the events, and the value of the element refers to the allocated timeslot. If the room allocation is also to be determined, a two-dimensional array is used in which the second dimension stores the room allocation decision. Examples can be found in studies by Corne, Ross, and Fang (1994); Ross, Hart, and Corne (1998); Deris et al. (1999); Perzina (2007); Yang and Jat (2011). The main issue with direct representation is that the ordinary crossover and mutation operators tend to generate illegal solutions. For instance, some timeslot are assigned with many events but others are assigned none. One needs to repair the offspring before it forms the next population, e.g., collect the illegal events, form them into a list with some order, assign them to available timeslots so far.

For permutation representation, no direct information of the assignments is encoded in the chromosome. Instead, it includes a permutation of events as the input for a decoder to transform into an actual solution. The decoder is often a greedy “scheduler”, which sequentially assign events into timeslots (and rooms if required) based on some rules. For instance, the first element (e.g., event) in the chromosome would be the first one to be processed by the decoder to generate a corresponding assignment. As a result, the search

²² In this thesis the reproduction operators refer to the genetic operators such as crossover, mutations, etc. It does not refer to the process to copy/survive individuals from generation to generation (Falkenauer 1997).

space highly depends on the decoder and may not reflect the whole actual search space. Using this indirect representation requires different reproduction operators to keep the order information to the offspring. As for the crossover operator, several different methods are used, and we describe two of them: The first one is partially mapped crossover (PMX) (Falkenauer 1997). Two cut points are randomly selected and the strings in between for two parents forms the mapping section. This substring is exchanged in the offspring, and the rest of elements are filled up with an element from its parent according to the mapping. The second one is uniform order-based crossover (Davis 1991). In this modified operator, two parents are filtered by a generated bit string template (0/1 mix rate is fixed) with the same length, i.e., positions from first/second parents. Meanwhile, the template that shows 1/0 are kept into the offspring. The remaining spaces in the first offspring are filled by the ones from the second parent. Second offspring is generated similarly. As for the mutation operator, scramble sub-list mutation is commonly used (Davis 1991). It selects a random-length sub-string in the parent chromosome and permutes it into a child. The number of infeasible solutions generated in the reproduction phase may be reduced, as the decoder specifically handles the infeasibility now. The solution quality may also be improved as the decoder can at least greedily consider the objective function. Besides the aforementioned incompleteness of search space, another main issue related to the permutation representation is the encoding redundancy. For instance, altering the order of the first three of four elements in the chromosome may not yield a different solution as the decoder will still assign them into the most

prioritized timeslot. In the extreme case when we only focus on the graph coloring problem, the redundancy is high as we are only interested in the grouping rather than the exact assignment from the group to the timeslot. In addition, reproduction operators may not yield different offspring either. For instance, mutation operators in the reproduction phase may need to generate a long and very different sub-string to keep diverse.

For grouping GA representation, the encoding is “group” oriented, which is different from previous two ways of representation. Every element in the chromosome stores the information in terms of grouping of events rather than a single event. It can be implemented by a variable-length array, which was introduced in Grouping Genetic Algorithm (GGA) by Falkenauer (1997). This implementation provides original encoding structure which merely considers the timing assignment (Eiben, van der Hauw, and van Hemert 1998; Erben 2001; Agustín-Blas et al. 2009). Another very popular form is by using a matrix (Lewis and Paechter 2005; Lewis and Paechter 2007). A row represents one room, and a column represents one timeslot. In this case, reproduction only works on groups, i.e., columns. It should be noted that the major difference between the previous two representations is not only in the encoding itself but also in the design of the reproduction operators. In GGA, both crossover and mutation operate on groups rather than events. On the other hand, one can always use the direct encoded chromosome to derive the grouping information first, and then apply the grouping GA operators in order to adapt the traditional encoding to a GGA. Various GGA crossover designs

have been proposed and most of them follow the scheme stated by Falkenauer (1997): Random crossing sections are firstly selected for both parents, where a crossing section reflects some of the groups in one parent. To generate the first offspring, we inject the crossing section of parent 2 before the position of crossing section of parent 1. We then eliminate the duplicated events in the original parent 1. As a result, some groups from parent 1 may be completely eliminated or left with only a few events. Depending on the objective function (e.g., minimizing the number of timeslots), such timeslots with too few events may be of poor quality, so an “adapting” process can be called which simply considers these events as unplaced (Falkenauer 1999). Adaption is usually implemented by some heuristic which assigns them back into existing groups or creates a new group with a better objective evaluation (Lewis and Paechter 2007). In this sense, adapt is similar to the heuristics used in permutation encoding. Mutation operator also works on groups/timeslots. Some randomly selected timeslots can be eliminated and the events associated could be assigned into some existing timeslots to create a diverse new solution. In addition, the group positions can be altered by invention, which reverses the groups within two randomly selected positions in the chromosome (Falkenauer 1999; Lewis and Paechter 2005).

Moreover, GA may overlook the exploitation of the search and may not dive deep enough to find a good solution. We describe two methods that GA can combine with to resolve such limitations, namely guided search and Variable Neighborhood Search (VNS).

Guided search is an obvious choice for GA because one can enforce the good parts of the population to survive in the next offspring, which is discussed by Yang and Jat (2011). The authors kept extracted information from the population on those events with zero penalties²³ associated and the corresponding timing and room assignments. A child was produced by either a normal crossover operator or a special constructive process which used the extracted information.

VNS can be incorporated with GA to increase the generality of the method. In studies by Burke, Eckersley, et al. (2010), GA was used to search the order of the neighborhoods in one run of VNS (i.e., try all the neighborhoods). Therefore, the chromosome was defined as a permutation of neighbors.²⁴ An initial generation of populations was generated with either greedy or random methods, and the corresponding solutions can be found after applying local search on each neighborhood according to the permutation within the VNS. The fitness of a chromosome was defined as the improvements between the solution after applying the current chromosome ordering and best solution in the initial generation. With the fitness evaluation, GA used the proportional selection and ordinary reproduction operators to generate a new permutation of neighborhoods.

²³ The penalties are in terms of both hard constraints and soft constraints.

²⁴ In that study duplication of neighborhoods is allowed in the chromosome but is essentially ignored when calculating the fitness value (and applying within the VNS).

From our observations and experiments, we find that it is hard to directly use the GA to solve our overall problem, which is still a MIP. Specifically, we find that the set of constraints make GA generate too many infeasible solutions from steps of population generation. In addition, reparation method seems difficult to be effective as there are usually too many violations presented. A preliminary test showed that GA is not suitable to solve our overall problem. In other words, GA is still not good enough to provide good feasible solutions for our overall problem. On the other hand, GA shows some potential to solve small scaled MIP problem (i.e., the number of constraints are relatively small) and the problem structure can be exploited. In our study, we use GA to solve a sub-problem of the overall problem, namely the core of the multi-dimensional knapsack problem. In that problem, the number of constraints is reduced very much. Also, the “matrix” of each “item” is non-negative, which means the chromosome representation of GA can be designed as a permutation of items while the solution can be constructed by simply packing items as long as no violation is incurred in every dimension. Moreover, we use the dual values (of each item) of the LP relaxation to provide additional evaluation on the population as the guidance of search.

Hyper-heuristics

Hyper-heuristics have attracted much attention in the field of timetabling in recent years. The motivation of using hyper-heuristics is that the “simple” meta-heuristics tend to require some level of detailed problem-specific knowledge in order to achieve promising results for some special kind of

problems. As a result, hyper-heuristics improves the generality for solving a range of timetabling problems. A hyper-heuristic utilizes more than one low-level heuristic during the search process to cope with different problem settings. From one iteration to another, a high level heuristic manages the calls for either one low-level heuristic from the heuristics pool or a combination of several low-level heuristics, and the management rules are usually based on the historical performances of the low-level heuristics. In some studies, once a good solution is found in some iteration, a local search was performed in order to find an even better one (See, e.g., Burke et al. 2007; Ersoy, Özcan, and Uyar 2007). The stopping criteria are usually the computational time or the number of runs. The main feature in hyper-heuristics is that the search space is based on the performance of low-level heuristics rather than the actual solutions. In our study, we use this idea of the hyper-heuristics by employing a two-approach method to generate a module selection in Chapter 4. Specifically, we develop a mechanism to intelligently select one of the two approaches in each iteration. Nonetheless, in the following, we summarize the application of hyper-heuristics in UCTP field and the two different main designs from the literatures.

Hyper-heuristics have been applied to several general and relatively easy timetable problems to raise the generality of the search methodologies. For instance, many studies on hyper-heuristics have considered Uncapacitated Exam Timetabling Problem (UETP) (Pillay and Banzhaf 2009; Qu, Burke, and McCollum 2009). In UETP, The hard constraints include HC1 and HC7, and

the soft constraints may vary slightly from one instance to another, but commonly include constraints like SC2. The word “incapacitated” means that there are no hard constraints such as HC2, HC3 or HC4. In such case, both hard constraints and soft constraints become less complex. For this reason the low-level heuristics could be easily developed and coded.

Typically, the low-level heuristics are usually those light and simple ones. The way that how the high-level heuristic calls low-level heuristics in the candidate solution generation phase can be divided into two approaches. The first one is called improvement approach.²⁵ This approach iteratively improves candidate solution, and in each iteration it generates a complete solution using only one low-level heuristic by selecting from a solution pool in one iteration. The second one is called constructive approach²⁵. It interactively constructs candidate solutions, and in each iteration it uses a permutation of heuristics from the pool rather than one specific heuristic, and each heuristic derived by a specific element in the permutation only generates a partial solution. In either case, if the generated solution is infeasible, a reparation method is called, e.g., back track techniques, heuristic to penalize the rank of the problematic (Qu, Burke, and McCollum 2009), tabu heuristic (Burke et al. 2007).

²⁵ The naming follows that by Burke et al. (2007). For the improvement approach, it is also named by researchers as *moving approach* or *perturbative approach* by Qu, Burke, and McCollum (2009).

In the improvement approach, a low-level heuristic is firstly selected by a select method, and the candidate solution in the last iteration is applied by this selected heuristic. Then the new solution is generated by applying an acceptance criteria. In the following, we summarize related studies in the field of exam timetabling which is in general very similar to course timetabling. We describe (1) the low-level heuristics; (2) the select method; (3) the acceptance criteria.

The low-level heuristics is often based on two types of neighborhoods: The first type is a simple move or exchange operation. An exam is randomly chosen and moved to a random²⁶ timeslot, or two exams are randomly chosen and their assigned timeslots are exchange, or two exams are randomly chosen and moved to a new timeslot independently. The second type is an improving move or swap operation: The move or swap has an aim to improve the fitness function based on the violation improvement of one set of hard/soft constraints. Move operation can change grouping of exams while swap operation additionally change the “consecutiveness” between groups (Burke, Eckersley, et al. 2010). Besides, there are other types of neighborhoods with special purpose. One of them is a Kempe chain swap, which specifically ensures that the swaps are feasible and can find a new move after applying previous two types of heuristics alone. As more exams can be swapped at the same time, the neighborhood could be further explored (the trade-off is the computation time).

²⁶ As usual the randomness here can refer to a pure uniformly random or randomness based on a tournament strategy.

See studies by Thompson and Dowsland (1996); Burke, Eckersley, et al. (2010).

There are plenty of select methods and acceptance criteria used in the literatures. Table 2-2 summarizes the select methods which are commonly used in the literatures and Table 2-3 summarizes the acceptance criteria.²⁷ With respect to the performance of improving approach, tabu search and choice function as well as improving and equal criteria are reported to have the best performance (Burke, Kendall, and Soubeiga 2003; Bilgin, Özcan, and Korkmaz 2007).

²⁷ Besides the listed ones in the table, some other select methods are sometimes used in the literatures, including: Case-based selector (Burke, Petrovic, and Qu 2006), Reinforce learning forced selector (Nareyek 2003), SA (Dowsland, Soubeiga, and Burke 2007) and GA (Han and Kendall 2003. Note that in this article a chromosome represents a list of low-level heuristics to call, i.e., it selects a heuristic list rather than a specific one). Other acceptance criteria include: Great Deluge Algorithm (Kendall and Mohamad 2004).

Table 2-2 Common select methods in the improvement approach

Simple Random	Choose randomly at a time (in only one iteration).
Random Descent	Choose randomly once but apply it in the following iteration as long as it improves the solution.
Random Permutation	Generates a permutation of heuristics initially, and apply each heuristic in each iteration.
Random Permutation Descent	Similar to Random Permutation, but apply a heuristic repeatedly if the improvement shows.
Greedy	Apply all the heuristics in each iteration but choose the one producing the best solution. For studies on methods from Simple Random to Greedy (Bilgin, Özcan, and Korkmaz 2007).
Choice Function	Different from previous methods, this method considers diversity. For each heuristic, its overall improvements and recent improvements ²⁸ in the historical calls and the time elapsed since this heuristic was last called jointly form a weighted sum function as the choice function. As a result, a descent search is focused when the improvement is large ²⁹ , and a diverse search is focused when the improvement is low. Various methods have been proposed to automatically determine the weights See, e.g., Cowling, Kendall, and Soubeiga 2001; Kendall, Soubeiga, and Cowling 2002).
Tabu Search	Low-level heuristic is ranked and selected to call. After the call, its rank will be increased only when a positive improvement is made in the derived solution and decreased otherwise. In the latter case, current selected heuristic is inserted into the tabu list, and other heuristics already in the tabu list is released if the improvement is negative. The tabu list plays a role to prevent a heuristics that performs bad recently to be applied again too soon (Burke, Kendall, and Soubeiga 2003).

²⁸ The idea of considering both historical performance as well as recent performance is very similar to the idea of exponential smoothing, and it can be deduced that this part of evaluation is iteratively predict the future performance of the low-level heuristic.

²⁹ A *large* improvement means a large positive improvement value of the new solution derived by the low-level heuristic. A *low* improvement means a small positive or even negative improvement value.

Table 2-3 Common acceptance criteria in the improvement approach

All Move (AI)	All new solutions are accepted (Cowling, Kendall, and Soubeiga 2001).
Only Improving (OI)	Only the one which improves candidate solution is accepted (Cowling, Kendall, and Soubeiga 2001).
Improving and Equal (IE)	Only the one which does not worsen the candidate solution is accepted (Bilgin, Özcan, and Korkmaz 2007).
Monte Carlo (MC)	OI + also randomly accept the non-improving new solution (Ayob and Kendall 2003).

Different from the improvement approach, a solution is built incrementally in the constructive approach. An initial solution is generated first followed by a local search. This approach can also be used when a partial solution is given and it constructs the undetermined part. In many cases, constructive approach determines an order of the decision points during the construction and then applies some allocation rules in each decision point.

Constructive hyper-heuristic approach utilizes several constructive heuristics as the low-level heuristics. It combines them by executing sequentially with a specific call sequence. Each call considers those decision variables not determined yet and determines one or several of them before passing the newly constructed partial solution to the next call. This method iteratively explores the search space of call sequences. Therefore, two slightly different call sequences could generate two very different solutions. In the following, we describe three key elements in this approach: (1) the low-level heuristics; (2) the neighborhood of call sequence; (3) the local search methods.

Among those constructive low-level heuristics, graph heuristics is commonly used especially when to solve exam timetabling problem. This is because this type of problems can be reduced into a graph vertex coloring problem where the vertex refers to the exam and the edge to the confliction. The graph heuristics determines an order of exams according to specific rules. After the solution is generated, a local search is sometimes implemented. Typical variants of graph heuristics are listed in Table 2-4. Specifically, the third column of the table indicates whether the ordering for the unscheduled exams is updated every time an exam is constructed into the solution. Among these methods, it is reported by Burke et al. (2007); Pillay and Banzhaf (2009) that LSD was able to generate comparably better feasible solutions. Therefore, a sequence concentrating on LSD is often used as the initial call sequence. To break a tie, these methods, except RO, need to introduce some randomness (Burke, Qu, and Soghier 2012).

Table 2-4 Comparison of commonly used graph heuristics in solving UETP

Heuristics	Sorted by	Update on order
LD: Largest degree	Conflict	Static
LCD: Largest color degree	Conflict unscheduled	with Dynamic
LSD: Least saturation degree	Feasible Timeslot	Dynamic
LWD: Largest weighted degree	Conflict /w Enrolment	Static
LE: Largest enrolment	Enrolment	Static
RO: Random ordering	/	Dynamic

The initial call sequence can be generated randomly or based on some prior knowledge. For instance, in studies by Pillay and Banzhaf (2009), exams

were ordered according to the Pareto comparison on the scores derived from different hybrid graph heuristics. The neighborhoods of call sequence varied from literature to literature: exchange position of two randomly chosen heuristics (Burke et al. 2007); change the heuristics to a random one in n randomly chosen positions (Qu, Burke, and McCollum 2009) or n randomly chosen consecutive positions (Qu and Burke 2005). Besides these simple moves, one can also embed with a VNS framework with different strategies, e.g., descent-ascent, biased one which focuses on exams with highest penalties (Qu and Burke 2005), or tabu search (Burke et al. 2007; Burke, Qu, and Soghier 2012). The occurrence of a low-level heuristic in the call sequence may be used to assign only one exam (Qu, Burke, and McCollum 2009) or several ones (Burke et al. 2007) to the timeslots. In addition, even the same sequence may yield different timetables in different runs, since there may exist multiple feasible timeslot assignments.

As for the local search, it tends to help reach those solutions that cannot be found by hyper-heuristics. The most popular local search method implemented in a constructive approach is deepest descend search that moves events to other timeslots as long as an improvement can be obtained (Burke et al. 2005; Burke et al. 2007).

With respect to the performance of the constructive method, a mixture of results is reported. It is found that the performance depends on many factors including the number and the composition of the low-level heuristics, the usage of RO or similar randomness ordering scheme, the usage of local search,

the conflict density³⁰ in the dataset, etc. In general, the constructive method is able to obtain sound results for many problems, including those hard problems (comparing to the results obtained by specifically developed algorithms).

In our study, it is challenging to use the hyper-heuristic approach to solve the overall problem directly. The main reason is that it is challenging to develop a simple heuristic to solve the overall problem by computing sufficient feasible solution of good quality. Therefore, combining multiple heuristics are even more challenging because it is also required to understand the strength and weakness of each individual heuristic. However, we adopt this idea by combining different approaches to solve a sub-problem in Chapter 4. We develop two sub-approaches to solve the module selection problem. The two approaches (both solve a multi-criterion problem. One focuses on computational efficiency and the other on solution quality) are combined in such a way that different iterations in the overall framework may call one approach or the other. The way of selection is similar to the aforementioned choice function and tabu search. Particularly, diversity is maintained by giving more chances for the approach which focuses on efficiency. The current sub-approach may be repeatedly executed until the assessment of solution quality shows a deterioration to some extent.

³⁰ It is defined as the ratio of the number of exams in conflict to the total number of exams.

Chapter 3 Data Analysis and Problem Modeling for MRPT

3.1 Overview

We first describe the problem overview of MRPT. A new campus is planned and built. Unlike the usual practice that some faculties are moved to the new campus, stakeholders define the intention of the new campus in this study as to create an environment which can nurture creativity, innovation and enterprise. The courses offered on the new campus is planned to be chosen from existing ones (that are offered in the original campus), which means a reallocation of modules is needed. One of the key requirements for the new campus is to involve students with diverse backgrounds so that different ideas can be openly exchanged. Hence, the stakeholders set a target to assign courses such that a good distribution of students from different disciplines can attend classes there. Moreover, the stakeholders would like students to enjoy the facilities on this new campus as early as possible. Therefore, it is preferred that the first-year and second-year students (i.e., junior students) become the majority of students who take courses on the new campus. To ensure a vibrant environment, the stakeholders also set a minimum requirement on resource-utilization on the new campus.

On the other hand, the stakeholders do not want to change the timetable from the current one for the practical reasons. These reasons are summarized in the following. Typically, variant curriculums exist at different

schools/departments in the medium and large university. As a result, there exist many intangible and unquantifiable constraints that we cannot capture; on the other hand, the schools/departments usually have built and deployed their preferred timetable for the sake of flexibility. Any changes to the existing timing may hugely disrupt the current educational activities and every faculty may be affected. As a result, the stakeholders collect the corresponding information on the current timetable from individual school/department and provide us one complete piece of timetable for all related modules. This timetable is then strictly followed in our study on MRPT.

The aforementioned considerations by stakeholders add a new dimension of complexity in the university timetabling problems. This is because that frequent commuting between two campuses by students who are heading for their classes is expected. We call the corresponding traffic (flow) between campuses the inter-campus traffic (flow). As the main commuting way to transport students between campuses is the shuttle bus service, if the module reallocation is not done properly, the potential inter-campus traffic flow can be so high that the shuttle service might not be able to handle the load.³¹ Moreover, the students might be late for classes. As a result, in MRPT, the stakeholders hope to identify the modules to be assigned to the new campus which can handle the high traffic flow while ensuring a good

³¹ Timetable can also affect the traffic flow. However, as mentioned already the timing is fixed in MRPT.

distribution of students across faculties, a high proportion of the first-year and second-year students and meeting a minimum resource utilization requirement.

To model MRPT, we first study the data given by stakeholders. Specifically, the key information we investigate include that how students register for modules, which school/department every student comes from and which admit term every student belongs to. We use historical student records to obtain the aforementioned data. These records also help us to predict the potential student registration for each module, as well as the relationship between two modules (i.e., the number of common students who register for both modules). This relationship can help us to estimate the potential student movement between the two courses when the timetabling schedules are given. With such information we learned, the traffic flow can be predicted under some assumptions, and the requirements set by the stakeholders can be specified into constraints with parameters.

In the next section, we discuss how we analyze the data and show the useful information we learned from his data analysis. In Section 3.3, based on the results from the data analysis, we introduce the mathematical programming formulation for MRPT. In Section 3.4, we conduct a series of numerical experiments to solve the mathematical model by commercial solver. The problem instances used are from the real data. In Section 3.5, we discuss the computational performancen of the results from numerical experiments conducted in previous, and highlight the limitation from solving MRPT by using the commercial solver. This motivates our further study in Chapter 4.

3.2 Data Analysis

Before we model MRPT, we first need to look at the data and obtain an overview of this problem. In addition, we need to determine the corresponding parameters that are used to do the problem modeling, such as student enrolment for each module and the student movement between modules. However, in the actual practice, the schedule and the venue need to be fixed before the student registration, so it is impossible to have the actual parameters before we solve the model. As a result, we propose to use the historical student registration records to estimate these parameters. We believe that this historical data will provide sufficient accuracy, and the main reason is that many of these modules are compulsory modules, especially for the first-year and second-year modules. Students have to select a fixed set of modules from semester to semester following the requirement set by their department and faculty. Since the student intake of the department on each year does not vary a lot, the enrollments for these compulsory modules are expected to be stable. For those non-compulsory modules where students can freely choose, even though their enrollment number might not be as stable as the compulsory modules, we expect that they will not impact negatively on traffic movement in our study. This is because most of the students are rational, and they will not choose back to back modules (modules consecutive conducted) to inconvenient themselves if these two modules are located on different campuses.

By using certain data analyzing techniques, we can obtain the module information from the data: the timing of each class for all modules, the venue where each class is held and the faculty which each module is offered by. Moreover, we can also obtain the student information: the faculty and admit term that a student belongs to and the modules that a student takes. From these pieces of information, we can compute some statistics (parameters) such as the number of students registered for a module and the proportion of students across faculties as well as first-year and second-year students taking each module. In the remaining part of this section, we list some of key statistics for one specific semester in our data as an example. The means to obtain such statistics and the insights that these statistics may provide are also discussed. Numerical experiment in Section 4 will also use this piece of data. In the following, we first show an overview on this piece of data. By conducting the data analysis, we then discuss how we help the stakeholders to determine the parameters with regard to their three main requirements, followed by our method to understand the nature of the inter-campus traffic by looking at the correlation among modules.

There were a total of 10173 undergraduate students, 402 modules offered by six main faculties after we have performed some preprocessing on the data.³² Table 3-1 shows the distribution of modules in terms of the module

³² Only undergraduate students are considered in this study, which is treated as a pilot study. Other students, such as graduate students, are planned to be considered in the future.

size (the student enrolment for each module). It shows that about 43% of modules had a class size less than 50 students, and only about 13% of modules had more than 250 student enrolments. Many small modules were either elective modules or modules of higher levels which were usually taken by senior students when they wanted to be specialized in a certain area. For those big modules, there were either general education modules or compulsory (fundamental) modules which are mainly taken by lower year students. Note that the class size (the number of students attending one lecture/tutorial) may be smaller than module size. This is because the capacity of the classroom was limited. For example, the student enrolment for first-year compulsory engineering modules can be as high as 1500 students, while the largest lecture room can only occupy 600 students. In this case, the attending students were divided into several classes, and they can be held at different times or in different rooms.

To help the stakeholders decide the parameters with regard to their requirements (e.g., faculty fairness, junior student priority and room utilization) about the new campus, we conduct various detailed analysis. First, we investigate the distribution of module size by the offering faculties. This information is also presented in Table 3-1. We find that the three major faculties, which are the faculty of engineering, the faculty of science and the faculty of arts and social sciences, contributed about 80% of the total modules offered in the university. Among these three faculties, faculty of engineering offered relatively fewer big modules. However, from Table 3-2, the

distribution on the student-module count shows that engineering students contributed the most.³³ It is because engineering curriculum required the students to take basic science and mathematics modules as compulsory modules. On the other hand, although the faculty of arts and social sciences offered many modules, a lot of these modules were also taken by other students because the university adopts a broad-based education. As a result, the student module proportion on the faculty of arts and social science was relatively smaller.

Based on these findings (and attitudes by the three main faculties), the stakeholders decide to set a guidance distribution as (30%, 30%, 30%, 10%) for faculty of engineering, faculty of science, faculty of arts and social sciences and other faculties.

³³ The student module count counts every student-module pair which representing students' module selections. For instance, a student attends three modules per week. For each of the three modules, his attendance contributes one student-module count.

Table 3-1 Number of modules and % of offering faculties grouped by module size range

Module Size Range	Number of modules	% from Engineering	% from Science	% from Arts and Social Sciences	% from Other faculties
0-50	171	30%	24%	28%	18%
51-100	92	30%	22%	29%	19%
101-250	87	17%	32%	28%	23%
251-600	43	20%	31%	32%	17%
> 600	9	20%	33%	31%	16%
Total	402	26%	26%	29%	19%

Table 3-2 Distribution of student-module count w.r.t origin of faculties

Engineering	Science	Arts and Social Sciences	Other faculties
33%	29%	28%	10%

Table 3-3 Distribution of student-module count w.r.t student grade

1st year student	2nd year student	3rd year student	4th year student
33%	30%	23%	14%

Similarly, we analyze the distribution of student-module count by analyzing the admit terms of students and grouping them by grades (e.g., first-year student, second-year student). The portion of junior students was currently 63%, partially because the university we study increases the enrollment of new students from year to year. As stakeholders specifically want to specifically attract junior students to enjoy the facilities on the new campus, they decide to set a guidance ratio as high as 80%.

The level of the minimum requirement for room utilization on the new campus is determined in a more complex way. A range of classrooms is offered on the new campus from small tutorial room to large lecture theatre.

On one hand, high utilization is preferred by stakeholders especially for those large rooms because the expensive facility and equipment can be properly utilized. On the other hand, setting a very high utilization level results in a large number of modules to be allocated, and its large module size typically leads to a high inter-campus traffic flow. In other words, the determination of this parameter partially depends on the solution to MRPT. To help stakeholders to set a promising level, we first define the measure of utilization of rooms as the weekly occupational hours. Then, we use part of the mathematical model developed in the next section to conduct scenario analysis. The model does not consider the inter-campus traffic, but considers the criteria related to resource utilization of different levels. With results generated from different scenarios, we are able to present the stakeholders with the potential of highest room utilization that can be achieved and the corresponding inter-campus traffic (The computation method is described later.). By carefully evaluating these results, the stakeholders decide that large classrooms should be highly utilized, although the resulting impact on inter-campus flow could be high. The small classrooms, however, is not needed to be highly utilized. This is because from the results we show that the small rooms are relatively excessive. Stakeholders then realize that these rooms can also be the venues for students' activities to help increase the low utilization from ordinary educational usage. The detailed decision on the utilization requirement is described in Section 3.4.

Aside from helping the stakeholders to set the parameters, we also try to understand the inter-campus traffic. Intuitively the inter-campus traffic is contributed by students who move for their next classes. As the timing is given, the students' movement behaviors primarily depend on their module registration. In other words, to understand the traffic across campuses, we first need to understand the relationship between different modules. With such considerations, we first investigate the "overlap" between modules. We then use the overlap to address the relationship of modules by grouping them into clusters. We find that these clusters provide insights on understanding the inter-campus traffic. We describe these three steps in the remaining of this section.

The module pair overlap, or simply overlap, is computed by analyzing the enrollment data. The overlap $o_{i_1 i_2}$ for every module pair $\langle i_1, i_2 \rangle$ is obtained by counting the number of students taking both modules i_1 and i_2 . When the course pair has high overlap, it means that these two courses are highly related, and hence can be a potential high contributor for traffic: Those students specified by $o_{i_1 i_2}$ may all need to travel between campuses if their time schedules fall in the same day but they are held on different campuses. From our analysis, the data show that there are significant module pair overlaps, i.e., many modules are related regarding student registration. Among these highly related modules, we can further categorize the overlaps into three groups in terms of the nature of relationship, i.e., strong structure overlap, soft structure overlap and preference overlap. Strong structure overlap is for those

module pairs where both modules need to be taken together by students as compulsory modules, which are required according to the curriculum. Soft structure overlap refers to those registrations that can be counted toward their major requirements such as modules related to department's specialization or technical modules. Preference overlap refers to those module pairs, where at least one of them is an elective module which satisfies the general university requirement. Typically they are the modules other than their major requirement, e.g., general education modules, broad-based modules. Even by knowing this grouping, the overlaps in each group still vary significantly. Therefore, we also group the modules in terms of overlaps by using the following clustering method, and show the result by combining the overlap type information.

We use a method based on the idea of clustering to identify the modules which are highly related, i.e., groups of modules such that modules inside the group have high overlap values while modules between groups have low overlap values. Overlap now plays the role of distance, i.e., high overlap indicates short distance. Specifically, our method uses the idea of the hierarchical clustering method (Hastie et al. 2013), which is a typical variant of connectivity-based clustering methods. We adopt the ideas of connectivity-based clustering because the traffic is contributed by the actual number of students who take classes on different campuses, or simply by the value of overlap. For the same reason, we do not consider the distribution, density or centroid of the module groups, which are other major topics in the field of

cluster analysis. Our method is essentially a single-linkage clustering method, or closest-neighbor clustering. In other words, the distance between two clusters C_1 and C_2 is defined as $d(C_1, C_2) = \max_{i_1 \in C_1, i_2 \in C_2} o_{i_1 i_2}$. Given the initial clusters such that each module is a single cluster, the method iteratively generates merged cluster by picking two clusters to merge so long as their distance is no less than a given threshold value, λ , until no more clusters can be merged. The main difference from the original hierarchical clustering method is that λ in our case has a practical meaning to determine whether the ‘connection’ between modules is significantly high or not and is pre-determined by stakeholders. This helps us to do the merging in one run rather than to merge one by one. This method is summarized in Algorithm 3-1.

Algorithm 3-1: Module-clustering method

input Set $S = I$ (I : the set of modules), module pair overlaps $\{o_{s_1, s_2} \mid s_1, s_2 \in S\}$, the threshold λ and count $n = 1$.

- 1 repeat**
- 2** select s from S to construct a new module cluster C_n , i.e., $C_n = \{s\}$, set $S = S \setminus \{s\}$.
- 3 for all** module pairs $\langle s_1, s_2 \rangle$ such that $s_1 \in C_n, s_2 \in S$
- 4 if** $o_{s_1, s_2} \geq \lambda$ **then**
- 5** $C_n = C_n \cup \{s_2\}, S = S \setminus \{s_2\}$.
- 6 end if**
- 7 end for**
- 8** $n = n + 1$.
- 9 until** $S = \emptyset$

output all obtained C_n .

As an example, we apply this algorithm to our data, and the results show that we can obtain around 22 clusters when stakeholders set $\lambda = 50$ (i.e., any overlap between two modules which belong to different clusters is less than 49.). We find that those big clusters usually contain modules from more than one faculty, and those smaller clusters usually consist those modules from individual faculty. Inside these clusters, we can further identify the type of overlaps so as to draw some insights on why they are highly related. Figure 3-1 shows the largest cluster among the 22 clusters obtained by running Algorithm 3-1. For this cluster, it can be further divided into six sub-clusters using the Iterative Conductance Cutting algorithm (Kannan, Vempala, and Veta 2000). This algorithm hierarchically splits one cluster into two by finding the minimum conductance cut. The key idea of this algorithm is to make sure

that there are more inner-connections within a sub-cluster compared to the inter-connection between sub-clusters. From the result after running this algorithm, the formation of these sub-clusters reflects the actual situation in the University. Most of the sub-clusters have modules come from the same faculty. For example, modules in the sub-cluster 5 are all from school of business, and they are strongly linked because most of the modules are compulsory courses within the faculty and their overlaps all belong to the group of “strong structure overlap”. The sub-cluster 1 contains modules coming from faculty of science, faculty of engineering and school of business. The reason for this large sub-cluster is that there are many first-year engineering students need to take sciences and mathematics modules as their compulsory subjects. Moreover, a lot of science students taking some popular business modules. Hence, the three types of the overlap can all be found in this largest cluster.

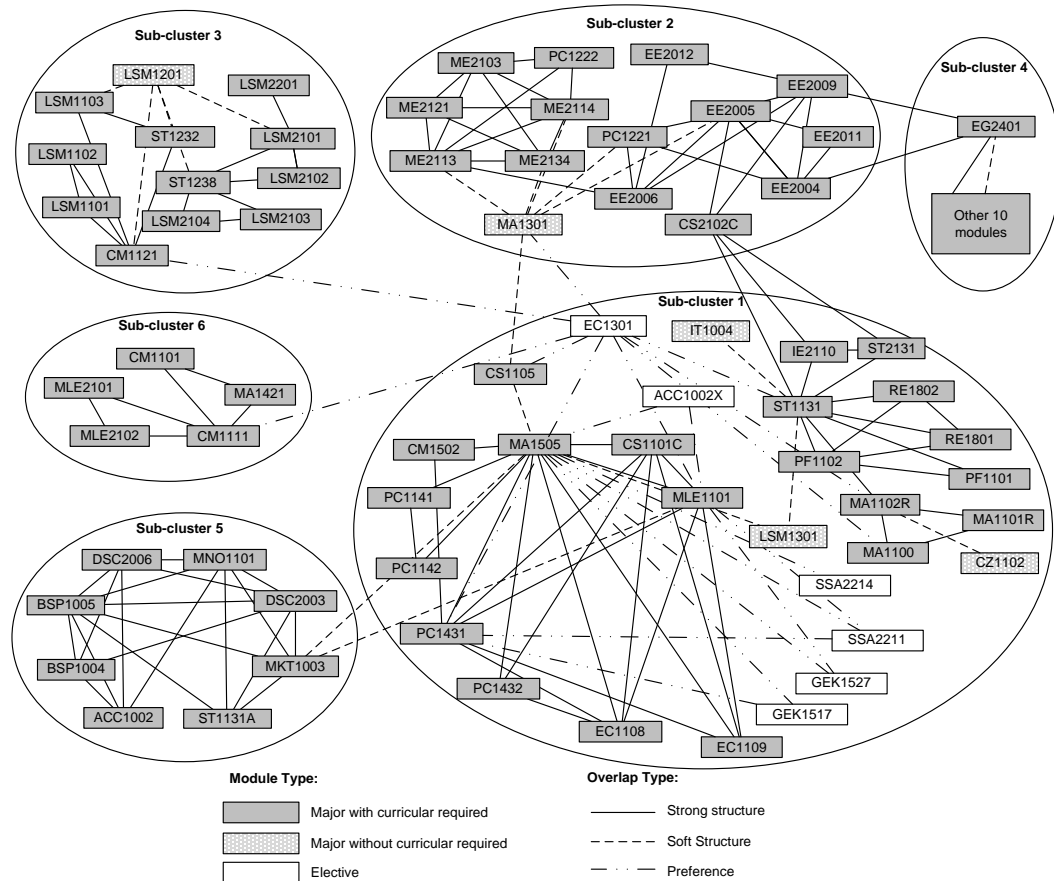


Figure 3-1 Results of sub-clusters³⁴ in the biggest cluster when $\lambda=50$

The results from clustering analysis indicate that there exist some groups of modules that within the clusters the correlations among modules are high. In addition, these groups share some common features, such as the similar offering faculties for the majority of the modules within the group. To optimize the inter-campus traffic, there may exist trivial solution such that assigning clusters to different campuses accordingly. However, the aforementioned requirements set by stakeholders prevent adopting these trivial solutions directly, as there are too many restrictions for a “cut” of clusters to

³⁴ Note that the tag of modules in this figure does not refer to any real ones.

satisfy. This finding indicates that MRPT is challenging. On the other hand, the clustering analysis provides us a way to find good solution indirectly with the following insight: The correlations of modules are primarily contributed by typically a subset of modules. Later, we are able to develop a surrogate measure of the inter-campus traffic. Details are discussed in Section 4.2.2.

3.3 Problem Modelling

With the preparation described in the last section, we formulate the MRPT as an MIP model in this section. We start from a brief description of MRPT.

We consider two campuses. Campus A is the existing campus, and campus B is the new extension. The aim is to identify modules to be taught at the campus B such that the traffic movement between the two campuses is minimized. We do not consider the traffic movement within the same campus as it has been taken care in the existing shuttle bus services. For the constraints, we consider the student proportion across different faculties has to follow certain target proportion, and also the first-year and second-year student proportion has to be at least greater than a certain target value. Note that for the target proportion values, we use the student-module count instead of using the individual student count because it is easier to implement in the model. On the new campus, there are different sizes of lecture rooms available. Since the student numbers for different modules are different, when we select modules to be assigned to the campus B, we will also determine which type of lecture rooms they should be allocated to. Note that although it is not possible to assign a big module to a small room, we will also want to ensure that no small

modules are assigned to big rooms so as the lecture rooms can be better utilized.

The following assumptions are made:

1. The students' movement behavior will not be affected by the module reallocation decisions.
2. We assume the length of each interval is 15 minute since we are measuring the student movement per 15-minute interval. We call it the student movement rate.
3. We do not consider the student movement for students attending the first lecture and students leaving after the last lecture in a day.
4. If a student needs to attend another lecture at the other campus, we assume that he will choose a time to travel to the other campus based on uniform distribution.
5. All lectures start at the hour and end at 15 minute before the next hour, e.g., 8AM to 9:45AM.
6. The class time can be from 8am to 8pm, Monday to Friday.

Notations

We list the parameters first, followed by decision variables. Among the parameters, we highlight two, namely traffic parameters, as both (each represents one direction) are important components to model the inter-campus traffic.

General parameters:

T : the set of one-hour timeslots in a week for timetabling. We only consider those timeslots in which inter-campus traffic will occur. For instance, suppose in every one of the five working days the first module in a day ends at 9AM and the last module in a day starts at 8PM, then 60 timeslots are needed and $\{t\} = \{1, \dots, 60\}$. For example, $t = 13$ refers to the 8AM-9AM slot on Tuesday in this case. In addition, we also use the index t to tag the last 15-minute period in its one-hour-long period, which is used to compute the traffic contribution.³⁵ For example, $t = 13$ also refers to the 8:45AM to 9AM period. Lectures and tutorials are all categorized as classes. A class may be short and occupy only one timeslot (such as a tutorial) or be long and occupy multiple continuous timeslots (such as a lecture).

I : the set of modules. Each module i may have several lectures and maybe even several tutorials associated.

J : the set of room types on the new campus. Each room type j has several rooms with the same or similar capacity and function. In general, lectures or tutorials of a lecture will be created in a similar size, making allocating all lectures or tutorials from one lecture to one single room type possible.

³⁵ The traffic contributions in other three 15-minute periods are no more than the last period, so we only need to focus on the last 15-minute period.

k_j : the number of rooms of room type j .

U : the set of faculties, e.g., Faculty of engineering, Faculty of science, Faculty of laws.

p_i : the number of students taking module i .

p_i^u : the number of students taking module i who are from faculty u .

q_i : the number of year 1 and year 2 students taking module i .

d_{it}^1, d_{it}^2 : the number of lectures/tutorials of module i that are conducted in timeslot t . Note that we allow multiple lectures/tutorials for the same module, which means for some $i' \in \widehat{I}$, there exists more than one $t' \in T$ such that $d_{i't'} \neq 0$.

Ω_j^1, Ω_j^2 : the set of modules whose lectures/tutorials are all compatible with the room type j , i.e., $i \in \Omega_j^1 / i \in \Omega_j^2$ implies all the lectures/tutorials of i have the proper sizes that will not exceed the room capacity and not waste spaces of type j . For instance, a module with 80 attenders is compatible with room types of 90 and 120 attenders, but not compatible with 150 attenders because the occupation ratio is considered too low.

e_i^1, e_i^2 : the number of timeslots that lectures/tutorials of module i use per week. We assume e_i^1, e_i^2 are all positive integers for all module. Therefore,

$$e_i^1 = \sum_t d_{it}^1, e_i^2 = \sum_t d_{it}^2, \forall i \in I.$$

Ξ_i^1, Ξ_i^2 : the set of matched room types for lectures/tutorials of module i .

g_u : the preference target ratio of faculty u in terms of fairness. For each faculty u , there is a preference of the ratio of the student module count from u on the new campus to the one from all the faculties. This ratio represents the level of involvement from a faculty, thus a large faculty may prefer for a higher ratio.

χ : the required minimum proportion of year 1 and year 2 student module count on the new campus to the overall one on the new campus.

ξ_u : the threshold used in the faculty fairness constraint. With it, we allow some mismatch between the target ratio of involvement from u and its actual involvement.

l_j : the number of timeslots per week that is required to be utilized for room type j . Normally the higher the capacity of the rooms is, the higher l_j is.

$v_{i_1 i_2}$: Auxiliary 0-1 variable. It is supposed to be 1 if module i_1 is reallocated, while i_2 is not.

Traffic parameters:

$r_{i_1 i_2 t}^1$: the traffic contribution (direction: from the new campus to the old campus) in the 15-min timeslot specified by t if module i_1 is reallocated (to

the new campus) but i_2 is not. It is computed through (1) computing the average student-movements at t in that direction caused by every possible class pair with respect to the module pair (i_1, i_2) , (2) Summing up all these averages. For instance, assuming, on Tuesday, module i_1 and i_2 has only one lecture and no tutorials respectively, although they may have classes on other days. The lecture of i_1 ends at 1PM and the lecture of i_2 starts at 3PM. 30 students take both lectures. Let t in this case representing the 1:45PM to 2PM period on Tuesday. Consider the case that i_1 is reallocated but i_2 is not. The average movement contributing $r_{i_1 i_2 t}^1$ at t from pair (i_1, i_2) is hence $30/9$ as there are nine 15-minute periods between the two lectures.

$r_{i_1 i_2 t}^2$: the traffic contribution (from the old campus to the new campus) at timeslot t if module i_1 is reallocated but i_2 is not. According to this definition, we have $r_{i_1 i_2 t}^1 = r_{i_2 i_1 t}^2, \forall i_1, i_2, t$.

Decision variables:

x_{ij} : the 0-1 decision variables representing whether the lectures of module i are assigned to room type j on the new campus. Denote set $X = \{x_{ij} : \forall i \in I, j \in J\}$. Note that lectures or tutorials from one module are allowed to be allocated to at most one room type.

y_{ij} : the 0-1 decision variables representing whether the tutorial of module i is assigned to room type j . Denote set $Y = \{y_{ij} : \forall i \in I, j \in J\}$.

Module selection decisions: The decision that which modules are reallocated

v_i : the 0-1 variable representing whether module i is reallocated to the new campus. This variable is defined only for simplicity. Denote set

$$V = \left\{ v_i, i \in I : v_i = 1 \text{ iff } \sum_j x_{ij} = 1 \right\}.$$

As a quick comment, $\{x_{ij}\}$ and $\{y_{ij}\}$ are called module reallocation decisions. They decide which modules are reallocated and which room type the corresponding lectures and tutorials are assigned to. $\{v_i\}$ are called module selection decisions. They decide which modules are selected to be reallocated.

Note that we need both $r_{i_1 i_2 t}^1$ and $r_{i_1 i_2 t}^2$ to capture inter-campus traffic of two directions. One may argue that many of them may be zeroes so that the complexity of the objective function is greatly reduced. This is partially correct. In reality there do exist several modules such that at most one class is held per day. For such module pairs, it is not possible to have traffic flow of two directions, and so for the corresponding $r_{i_1 i_2 t}^1$ and $r_{i_1 i_2 t}^2$, at least one is 0. However, in reality there are also many modules which may have multiple classes on the same day. For instance, one module is so large that students have to be split into two groups. The big scale of the school, the teachers' preference and the scarcity of the resources are possible reasons. Two of such modules may incur a traffic flow at some timeslots from both directions if they

are assigned to different campuses. In this case, both corresponding $r_{i_1 i_2 t}^1$ and $r_{i_1 i_2 t}^2$ are non-zero.

Also note that, if for any module there is at most one class held on each day, at least one from $r_{i_1 i_2 t}^1$ and $r_{i_1 i_2 t}^2$ is 0. However, this assumption is not true in reality especially for large university. It is possible that both $r_{i_1 i_2 t}^1$ and $r_{i_1 i_2 t}^2$ for specific i_1, i_2, t are positive. For instance, both i_1 and i_2 can have multiple classes on the same day of t , e.g., two lectures of i_1 are held to cater two groups of students, as well as for i_2 . Given the location of modules (but assuming they are on different campuses), it is possible that some of students from i_1 need to travel to attend class of i_2 , and vice versa. Therefore we can have traffic from both directions with respect to only one module pair. For this reason, we need both $r_{i_1 i_2 t}^1$ and $r_{i_1 i_2 t}^2$ to model inter-campus traffic.

With parameters in MRPT described, we discuss the objective function. This function is to minimize an inter-campus traffic measurement, which is the worst-case scenario of traffic rates across different time slots and directions. We denote this inter-campus traffic measurement as:

$$F(V) = \max_{t \in T} \left(\sum_{i_1, i_2 \in I} r_{i_1 i_2 t}^1 \max(v_{i_1} - v_{i_2}, 0), \sum_{i_1, i_2 \in I} r_{i_1 i_2 t}^2 \max(v_{i_1} - v_{i_2}, 0) \right) \quad (3.1)$$

In this measurement, $\left(\sum_{i_1, i_2 \in I} r_{i_1 i_2 t}^1 \max(v_{i_1} - v_{i_2}, 0) \right)$ represents the traffic

rate derived from new campus to old campus at timeslot t , and

$\left(\sum_{i_1, i_2 \in I} r_{i_1 i_2}^2 \max(v_{i_1} - v_{i_2}, 0) \right)$ represents the one of the other direction. (Note that

$\max(v_{i_1} - v_{i_2}, 0) = 1 \Leftrightarrow i_1$ is relocated and i_2 is not .) Both depend on the decisions of all modules, i.e., not only on those which are selected to be reallocated, but also on those which are not.

It should be noted that our traffic modeling has a limitation. The traffic measured in (3.1) might be higher than the actual traffic movement. This is because when we compute the student movement, we only use the first order information, i.e., the course pair overlap. However, it might be possible that there are higher order effects which can bring down the student movement. For example, a student is taking three modules in the same day, the first module is at campus A and the remaining modules are at campus B. After taking the first module, the student will move to campus B and spend his remaining time there to take the second and the third modules. In our model, by only considering the overlap, we will double count by considering both overlaps between module 1 and module 2, and between module 1 and module 3. Fortunately in the next section we can show that this higher order effect is small.

Because (3.1) is not in linear form, we linearize it with the following formulas by introducing real variable z and aforementioned auxiliary variables $\{v_{i_1 i_2}\}$ (detailed proof is shown in [A.3](#)):

$$\min z \tag{3.2}$$

Subject to

$$z \geq \sum_{i_1, i_2 \in I} r_{i_1 i_2}^n v_{i_1 i_2}, \forall n \in \{1, 2\}, t \quad (3.3)$$

$$\sum_j (x_{i_1 j} - x_{i_2 j}) \leq v_{i_1 i_2}, \forall i_1, i_2 \in I \quad (3.4)$$

$$v_{i_1 i_2} \leq \frac{1}{2} \left(1 + \sum_j (x_{i_1 j} - x_{i_2 j}) \right), \forall i_1, i_2 \in I \quad (3.5)$$

$$z \in \mathfrak{R}; v_{i_1 i_2} \in \{0, 1\}, \forall i_1, i_2 \in I \quad (3.6)$$

With a linear objective function, the MRPT can be modeled as a MIP problem **P1**:

$$[\mathbf{P1}] \quad \min z$$

Subject to

$$(3.3)-(3.6)$$

$$(1 - \xi_u) g_u \leq \frac{\sum_{i,j} p_i^u x_{ij}}{\sum_{i,j} p_i x_{ij}} \leq (1 + \xi_u) g_u, \forall u \quad (3.7)$$

$$\frac{\sum_{i,j} q_i x_{ij}}{\sum_{i,j} p_i x_{ij}} \geq \chi \quad (3.8)$$

$$\sum_j (x_{ij} - y_{ij}) = 0, \forall i \in I \quad (3.9)$$

$$\sum_j x_{ij} \leq 1, \forall i \in I \quad (3.10)$$

$$\sum_i d_{it}^1 x_{ij} + d_{it}^2 y_{ij} \leq k_j, \forall t \in T, j \in J \quad (3.11)$$

$$\sum_i e_i^1 x_{ij} + e_i^2 y_{ij} \geq l_j, \forall j \in J \quad (3.12)$$

$$x_{ij} = 0, \forall i \notin \Omega_j^1; y_{ij} = 0, \forall i \notin \Omega_j^2 \quad (3.13)$$

$$x_{ij}, y_{ij} \in \{0,1\}, \forall i \in I, j \in J \quad (3.14)$$

In P1, The decisions we make include (i) *modules reallocation*: whether or not to relocate the module i to the new campus ($\sum_j x_{ij} = 1 \Leftrightarrow$ Relocated and $\sum_j x_{ij} = 0 \Leftrightarrow$ Not relocated); (ii) *room assignments* : for reallocated module i , which room types should its lecture(s) and tutorial(s) (if any) be assigned ($x_{ij} = 1 \Leftrightarrow$ lecture(s) of module i assigned to room type j and $y_{ij} = 1 \Leftrightarrow$ tutorial(s) of module i assigned to room type j). Decision variables $\{v_{i_1 i_2}\}$ and z are then implied by these two decisions.

The constraints to satisfy include the followings:

(1) Faculty fairness: constraints (3.7) ensure that the distribution of students-module count from a specific faculty on the new campus is consistent with the desired distribution. The reason for this restriction is because in this problem all faculties have equal utilizing opportunity. The student-module count is one way to measure the contribution of faculties' involvements. Therefore, the distribution based on this measurement represents the fairness

of involvement among faculties. If the actual ratio for faculty u equals the

desired ratio g_u , then $\frac{\sum_{i,j} p_i^u x_{ij}}{\sum_{i,j} p_i x_{ij}} = g_u$. As an equality constraint would be too

strong in practice which can easily lead to infeasibility, we allow a range centered at r_u which is specified by ξ_u . As a result, these constraints can also apply to those situations such that there is a preference on faculty utilization distribution.

(2) Student preference: constraints (3.8) ensure that the first-year and second-year student proportion on the new campus is no less than the required level. To see this, $\sum_{i,j} q_i x_{ij}$ represents the actual 1st and 2nd student-

module count, and $\sum_{i,j} p_i x_{ij}$ represents the overall student-module count given the actual module reallocation. The main reason of setting this restraint is to promote future student enrollment. In general, it can refer to any constraints which prioritize a specific module group.

(3) Room assignment: It includes several parts. Firstly, it is not allowed to have lectures and tutorials to be on different campus, as typically the tutorial is scheduled right after the lecture. In addition, because of the similarity in size, all the lectures/tutorials of a module must be allocated to only one room type on the new campus if this module is reallocated. These requirements are illustrated jointly by constraints (3.9) and (3.10). Secondly, each room type has a limited number of rooms which restraint the number of

classes can be assigned per week, which is illustrated in constraints (3.11), and we call them *capacity constraints*. Thirdly, each room type should be utilized well, which is measured by its weekly occupied hour. The occupation hour in each room type shall be larger than the desired level, which is illustrated in constraints (3.12) and we call them *demand constraints*. Fourthly, the classes assigned should be compatible with the room types. Constraints (3.13) ensure this requirement.

We can conduct some redundancy check on the problem instance. We start from room assignment constraints. All coefficients in type 2 constraints are positive so that some variable fixing or elimination of constraints is possible. First we can reform (3.11) and (3.12) to the following inequalities:

$$\sum_i d'_{it} \rho_{ij} \leq k_j, \forall t \in T, j \in J \text{ and } \sum_i e'_i \rho_{ij} \geq l_j, \forall j \in J \text{ where } \rho_{ij} \text{ refers to any}$$

decision variables that is not fixed to zero due to (3.13) and d'_{it} and e'_i are its corresponding coefficients in (3.11) and (3.12). Then we check the validity of the following conditions: (1) $\sum_i e'_i > l_j, \forall j \in J$, (2) $\max_{i,t} d'_{it} \leq k_j, \forall j \in J$ (3)

$$\min_i e'_i \leq l_j, \forall j \in J .$$

If some is violated, either the corresponding decision variable can be fixed to zero, or the whole constraint is redundant. In the remaining part of this paper, we assume these conditions are all met. We then check type 1 constraints. Note that it is possible that some coefficient is negative, reflecting that selecting the corresponding module will move away from the target ratio. Similar redundancy check is hence not effective.

We briefly analyze the scale of **P1**. In total, there are $O(|I|^2 + 2|I||J|) \approx O(|I|^2)$ columns and $O(|T||J| + |U| + |I| + |I|^2) = O(|I|^2)$ rows, as normally $|I| > \max(|T|, |U|, |J|)$. In practice, $|I|$ can easily go up to hundreds, thus the scale of this MIP problem could be enormously large.

3.4 Numerical Experiments

We conducted several numerical experiments based on the data that is used in Section 3.2. Experiments were grouped by different constraint requirement, ranging from slack to strict. In addition, we examined the higher order effect mentioned in the ending of the last section is not severe.

In Table 3-4 we show five sets of scenarios based on different parameter settings. Scenario *A* was the reference case which was used in our real-world project. In this scenario, we controlled the faculty fairness parameters by setting three largest faculties to have roughly equal number of student-module counts in campus B, i.e., between 30% and 36%. We also controlled the first-year and second-year student-module count proportion to be at least 80%. For the room utilization requirement, we required at least 8 hours usage per day per room for big rooms; at least 6 hours for medium rooms; and no requirement for small rooms. In order to study the impact of the faculty fairness, in Scenario *B*, we relaxed the constraint (4). Similarly, we investigated the impact on the student preference constraint by relaxing constraint (5) in Scenario *C*. Scenario *D* and *E* referred to the impact on the

room utilization requirement by varying RHS of constraint (6). In particular, we looked into a medium and a low utilization requirement.

Table 3-4 Five scenarios and their parameter settings

Parameter settings	Scenario A	Scenario B	Scenario C	Scenario D	Scenario E
Faculty fairness	$\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$	-	$\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$	$\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$	$\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$
Junior student preference	$\geq 80\%$	$\geq 80\%$	$\geq 50\%$	$\geq 80\%$	$\geq 80\%$
Room utilization	(8, 6, 0)	(8, 6, 0)	(8, 6, 0)	(6, 3, 0)	(4, 1, 0)

Table 3-5 summarizes the solutions given by the solver for all five scenarios. They were obtained by running CPLEX 11 on an Intel Core 2 2.6GHz desktop. The code was programmed in C# by calling .NET CONCERT interface. We chose C# and API because it is also a convenient way to design and implement a user interface which the user should be more familiar with. In fact, the way of using CPLEX was identical throughout this work. The computational budget given to every scenario was about 1 hour. It should be highlighted that the solver in no scenario can obtain optimal solution within 1 hour. In fact, the actual performance (including solution quality, the number of feasible solutions obtained) was very poor. The details are shown in the next section.

Table 3-5 Result summary for 5 scenarios

Evaluation	Scenario <i>A</i>	Scenario <i>B</i>	Scenario <i>C</i>	Scenario <i>D</i>	Scenario <i>E</i>
OBJ: Objective value in P1	246.9	128.4	178.3	89.4	76.5
No. of modules reallocated	146	186	208	98	137

Scenarios *B* and *C* are relaxed problems of scenario *A* where we observe lower traffic movement and more modules are being assigned to campus B. For scenario *B*, it is found that modules from school of business, which was a relative small faculty, were mostly assigned to campus B. In fact, all modules from sub-cluster 5 in Figure 3-1 were assigned. Further analysis suggested that the assigning of these business modules to campus B would not affect too much on the traffic because these modules were mainly taken by their own students. For scenario *C*, the improvement in the objective value was not as significant as scenario *B*. We can explain this in multiple ways. First, many of the large modules were primarily first-year and second-year modules, and since we needed to have high utilization on the larger room, most likely these first-year and second-year modules would still need to be assigned to campus B even though we are able to relax the constraints. Moreover, in order to maintain the faculty fairness, many small modules which were taken by the higher year students would be forced to assign to campus B. Hence, in scenario *C*, the traffic movement was more than scenario B and more modules were assigned to campus B.

Another interesting observation is that more number of modules assigned to campus B did not always imply higher traffic. For scenario *B* and *C*, even though we had more modules assigned to campus B, the traffic was smaller. This can be explained by the fact that we can always assign the modules which have high overlap to the new campus together.

For scenario *D* and *E*, lower traffic was observed because we had lower room utilization requirement on large lecture room. In general, large modules can potentially contribute more traffic. In addition, since scenario *E* had the lowest utilization requirements on large rooms, it essentially had more options to select those big modules with lower movement rate contribution. However, it may violate other constraints such as the faculty fairness and student preference constraints. This can be resolved by selecting more small modules which we observed by comparing scenario *E* and scenario *D*.

Table 3-6 Result summary for 5 scenarios with other evaluations

Evaluation	Scenario A	Scenario B	Scenario C	Scenario D	Scenario E
OBJ: Obj value in P1	246.9	128.4	178.3	89.4	76.5
OBJ1: Traffic impact without using overlap	224.1	126.7	162.9	86.3	74.4
OBJ2: Traffic impact only considering back-to-back	219	124	156	85	74

In the end of this section, we show that effect of higher order effect.

Table 3-6 compares our model objective (OBJ) with the actual traffic (OBJ1)

which was computed by the student movement based on every student's timetable after the course assignment plan was given (which is the solution for OBJ). It shows that the difference between them was very small. A brief explanation of why the higher order effect was insignificant is that the condition to form a significant high-order overlap is typically strong. For instance, to form an overlap with order of 3, three modules which are strongly related are needed to be scheduled to the same day. Since a group of modules which are highly related are commonly offered by the same department, it is unlikely that the department arrange three such modules on the same day. The higher the order of overlap is, the lower the chance of scheduling on the same day is. It is hence expected that the actual traffic (OBJ1) is not over-estimated a lot. Furthermore, we also compared the model objective with the back-to-back traffic movement (OBJ2) for the assignment plan. The back-to-back traffic movement is defined as the number of students who need to travel to the other campus immediately after the end of the lecture because they are taking consecutive courses that are located on different campuses. The difference between OBJ and OBJ2 was very small, which indicates that the back-to-back traffic movement contributes to the high proportion of the worst traffic. The main reason for this insignificant effect is that, for every student, the average number of modules taken within a day is typically relatively small.

In general, our results influenced the stakeholders' decisions in real life in several ways. First, they used our results in planning the modules of the first year on the new campus. Although the results were not directly adopted as

there are some other considerations that the stakeholders must take in the end (which is beyond our territory) but most of the modules and room allocation are from our results. In fact, the final model we solved was very simplified in which only about 100 modules were to be selected. This was due to the fair amount of constraints addressed from various departments since they concerned that their own students may suffer from problematic traffic. However, this situation was resolved in the following years due to the success from the first year's running. Second, the traffic evaluation from our results also suggested that the current shuttle bus system could not handle the students unless investing more buses based on the computed traffic values(it also means that our model cannot decrease the traffic for ever in the presence of the set of constraints). As a result, more buses were later deployed to help ease the traffic pressure. Third, the best we could do at that time is using the historical data. Consequently the results shown by our model at that stage should be worse than the results from the real data, which were captured in the planning process of the second year on the new campus. It is reported that the results from the following year is more consistent with the real traffic.

3.5 Discussion

Several issues related to using solver to solve MRPT are observed and discussed in the following.

- (1) The unpromising performance of the solver on our test case:

The results of numerical runs show that CPLEX cannot find the optimal solution for any test case before it runs out of memory.³⁶ Numerical results also show that obtaining a relatively good solution ($\text{gap}\% \leq 15\%$) is not possible in 20 hours for all scenarios. To conduct further investigation, we create two new sets of experiments, and both have no limit in computation time. We first conduct experiments on all five scenarios in the default setting of CPLEX. On average, 8.2 feasible solutions are found and within which, there are only 1.1 solutions that are within 30% optimality gap. We then change parameters of the solver to focus feasibility, and the number of feasible solutions is increased by 240% on average. However, the number of good solutions rarely changed as only 2.2 solutions are obtained on average.

(2) The impact from the scale of the problem:

In the test case of our study, we use a scale of a moderate scale of university, and $|I|$ is several hundred and $|T|$ is 40. In a typical large university, the number of variables can be even more, e.g., $|I|$ goes up to thousands. In addition, the number of indicator variables introduced in P1 is $|I|^2 |T|$, bringing much more variables. Such a large number of variables has a significant impact on performance of conventional solution techniques, e.g., each LP relaxation problem requires a large amount of time to solve to optimality during the branch and bound.

³⁶ The way of test case generation is [described](#) in the appendix.

(3) Possible reasons for the aforementioned issues.

We investigate the problem structure. By analyzing the constraints part, we expect that finding a feasible solution of P1 is rather difficult. For instance, even if we fix the module selection, the leftover problem is a room assignment problem which tries to find a feasible room assignment for all selected modules. This assignment-type problem is a hard problem, because we show in 4.3.2 that it can be transformed into several knapsack-type problems, e.g., a constrained multiple multidimensional knapsack problem, or a multiple multi-dimension knapsack problem after relaxing several constraints. All such problems are well-known difficult ones.

Chapter 4 An Iterative Two-Phase Approach to MRPT

4.1 Overview

In Chapter 3, we find two main difficulties in solving **P1** (described in 3.5). The first issue is that the solver cannot find good solutions for our test cases. The second issue is that when the problem scale becomes larger, the solver may not even find a feasible solution. As a result, we develop a heuristic that overcomes the aforementioned difficulties.

We investigate the problem structure of **P1** and finds out that it is possible to categorize the decision variables and constraints into two groups, which are shown in Table 4-1:

Table 4-1 Groupings on constraints and decision variables

Constraint Categorization	Decision
Type 1 Constraints <i>faculty fairness and student preference constraints (3.7-3.8)</i>	Module selection v_i : Whether module i is selected to reallocate
Type 2 Constraints <i>room constraints (3.9-3.13)</i>	Room assignment (<i>depending on the module selection</i>) x_{ij} and y_{ij} : the room types assigned to the lecture/tutorial of module i

Under this grouping, we find that type 1 constraints only depend on module selection variables, while type 2 constraints only depend on the room assignment given by the module selection. We cannot determine the room assignment decision without determining the module selection decision first. In addition, the objective value only depends on the module selection decision.

As a result, we first ignore the room assignment decision by searching for a module selection that has a good objective value. After we obtain a set of modules that are to be reallocated, we can further determine their room assignment.

Motivated by this idea, we develop a sequential two-stage approach. The first phase is to find the modules that are to be allocated to the new campus and also satisfy type 1 constraints. Even though we do not consider the room assignment decision in this phase, we do consider type 2 constraints (which are related to the room assignment) implicitly. By doing so, we hope that the results from phase 1 can help us to find a feasible room assignment in the phase 2 easily. When a feasible module selection is found in phase 1, we proceed to phase 2 which is to find the feasible room assignment for these modules that satisfy type 2 constraints. If we can find such an assignment, the overall solution becomes a solution candidate for P1. Otherwise, we restart the algorithm. When a feasible module selection is not found in phase 1, we also restart the algorithm. Through this way, we iteratively repeat these two phases until the stopping criteria (time budget) is reached. The suggested solution is the best solution among those that are found so far. In addition, in every iteration, we learn useful information from the result of previous runs to adjust certain parameters for the future iteration. The process is shown in Figure 4-1, and the details on the two phases are described in the following.

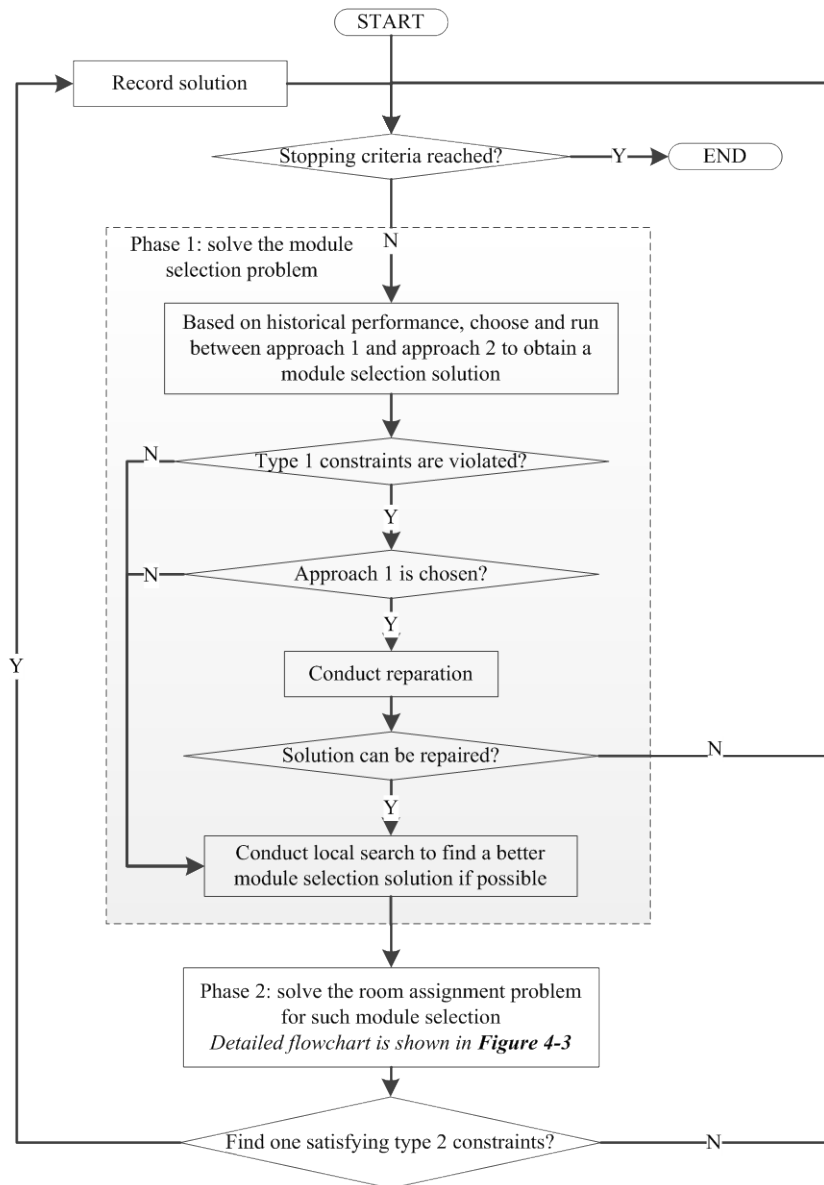


Figure 4-1 The overall framework of the proposed heuristic

In the first phase, we use multi-objective optimization framework to solve the module selection problem for two reasons. First, many type 1 constraints have negative coefficient (e.g., $(q_i - p_i)$ for $\sum_j x_{ij}$ in (3.8)). By

treating them as hard constraints, the constructive method may stop too soon

to find a feasible solution. Second, the violations of type 2 constraints have to be also monitored, and the typical way is to minimize their aggregated violation. Therefore, the multiple objectives consider the original objective, the violations of type 1 constraints and the violations of the aggregated measure of the type 2 constraints.³⁷ Under this framework, we propose two approaches to tackling the problem. In each iteration, we choose one approach from the two based on the results from the previous iteration.

In approach 1, we use a greedy constructive approach to select modules one by one. In each construction step, we first randomly select one objective from the multi-objectives discussed before. Based on the selected objective, we rank those unselected modules accordingly. We then randomly select one module among those good candidates according to their performances. The construction process is stopped when either a solution satisfying type 1 constraints is found or the solution shows no improvements in reducing violations after a number of consecutive steps. If we cannot find a solution satisfying type 1 constraints, we use a reparation method to repair the infeasible solution. After the feasible solution is found, a local improvement process is executed in order to further improve the objective value while

³⁷ As for simplicity, we say “aggregate type 2 constraints” by aggregating room capacity constraints (3.11) and room utilization constraints (3.12) respectively. For the former constraints, constraints of different room types and time slots are aggregated. For the latter constraints, constraints of different room types are aggregated.

maintaining the feasibility in terms of type 1 constraints. The details of the approach 1 are discussed in Section 4.2.1.

In approach 2, instead of using the constructive approach which selects module one by one, we explicitly build an MIP-based multi-objective model to select the modules. In the proposed MIP model, we specifically set two objectives. The first objective considers the inter-campus traffic by introducing a surrogate measurement based on the results from cluster analysis mentioned in chapter 3. The second objective considers the violations of a subset of type 2 constraints in an aggregated way. The type 1 constraints remain as hard constraints, while the rest of type 2 constraints are considered in such a way that the violation of the aggregated measure cannot exceed a certain threshold. We solve this MIP model by using the idea of NBI method which are summarized in Chapter 2. Similar to approach 1, if we can find one feasible solution, local improvement will be applied accordingly. The details of the approach 2 are discussed in Section 4.2.2.

In the second phase, based on the module selection obtained in the first phase, we try to find a feasible room assignment. We first formulate it as a MIP model and then use the branch and bound framework to solve this problem. We branch on the possible room assignments for a selected module. We use Lagrangian relaxation method to solve the sub-problem in the branch and bound framework, and use a primal heuristic to find a solution in case that the result from solving by Lagrangian relaxation method is not integral. Specifically, the dual bound and primal bound obtained help us to prune the

branch. When using the Lagrangian method to solve the sub-problem in the branch, we show that it can be decomposed into a collection of multi-dimensional knapsack-type problems, which give us an efficient way to solve the problem. The Lagrangian dual is solved by the sub-gradient search method. On the other hand, the primal heuristic is based on Constraint Programming (CP) technique. The details of phase 2 are discussed in Section 4.3.

4.2 Phase 1: Module Selection Problem (MSP)

The goal of the first phase is to identify a set of modules such that reallocating these modules satisfies type 1 constraints while having a good objective value. Type 2 constraints, however, are not ignored completely. This is because if so, there would be no control on the room assignment and it can be very difficult to find a room assignment for the resulting reallocation solution. Instead, we consider type 2 constraints indirectly by using aggregating method. We now define a module selection $V = \{v_i \in \{0,1\}, i \in I\}$, indicating whether module i is selected or not, which is a solution to MSP modeled by **P2**:

$$[\mathbf{P2}] \quad \min \max_{t \in T} \left(\sum_{i_1, i_2 \in I} r_{i_1 i_2 t}^1 \max(v_{i_1} - v_{i_2}, 0), \sum_{i_1, i_2 \in I} r_{i_1 i_2 t}^2 \max(v_{i_1} - v_{i_2}, 0) \right)$$

Subject to

$$(1 - \xi_u) g_u \leq \frac{\sum_i p_i^u v_i}{\sum_i p_i v_i} \leq (1 + \xi_u) g_u, \forall u \quad (4.1)$$

$$\frac{\sum_i q_i v_i}{\sum_i p_i v_i} \geq \chi \quad (4.2)$$

$$\sum_i D_i v_i \leq \hat{K}, \text{ where } D_i = \sum_{j \in J, t \in T} (d_{it}^1 + d_{it}^2), \hat{K} = |T| * \sum_j k_j \quad (4.3)$$

$$\sum_i E_i v_i \geq \hat{L}, \text{ where } E_i = (e_i^1 + e_i^2) \times |J|, \hat{L} = \sum_{j \in J} l_j \quad (4.4)$$

P2 shares the same objective function as P1. The decision variables in P2 are only $V = \{v_i\}$. Individual decisions for lectures or tutorials are not needed, because when we aggregate type 2 constraints, both decisions for module i refers to the same v_i due to the constrain (3.9) that requires that the lectures and tutorials should be reallocated at the same time for one module. In terms of the constraints of P2, (4.1) and (4.2) are essentially (3.7) and (3.8) (the faculty fairness and student preference constraints) on variables $\{v_i\}$. Constraint (4.3) and (4.4) can be considered as an aggregated constraints for (3.11) and (3.12). They relax original capacity restraint and room utilization requirement by aggregating corresponding constraints ($(|T|+1)|J|$ constraints are now aggregated into 2 constraints). They can be interpreted as the lower and upper bound of the number of modules selected. A solution to P2 fully satisfies type 1 constraints but may not satisfy all those type 2 constraints.

As P2 is still not easy to solve, we introduce the multi-objective framework by relaxing certain hard constraints and transform them into soft-constraints. Hence, the multiple objectives may include the original objective,

the violations of type 1 constraints and the violations of the aggregated type 2 constraints. Under this framework, we propose two approaches to tackling the problem, which are discussed in detailed in the following two sections.

In the initial of the algorithm with regard to phase 1, we run approach 2 for several iteration which is followed by approach 1 for another few iteration. After obtaining enough performance information of both approaches, we decide which approach to choose for subsequent iterations. The criteria that we use to evaluate the performance are similar to the four criteria which are used in approach 1. We compute the actual objective value according to (3.1) for results obtained from approach 2 as approach 2 uses the surrogate measure of traffic. We give higher chance to the approach that gives better overall performance.

4.2.1 Approach 1: Greedy constructive procedure

This method is based on the construction phase from the study by Martí et al. (2011) in the field of Greedy Randomized Adaptive Search Procedure (GRASP). For the current constructed module selection V , the following multiple objectives $\Lambda_1(V)$ to $\Lambda_4(V)$ are considered:

$$\Lambda_1(V) = \max_{t \in T} \left(\sum_{i_1, i_2 \in I} r_{i_1 i_2 t}^1 \max(v_{i_1} - v_{i_2}, 0), \sum_{i_1, i_2 \in I} r_{i_1 i_2 t}^2 \max(v_{i_1} - v_{i_2}, 0) \right)$$

$$\Lambda_2(V) =$$

$$\sum_u \max \left(\max \left(\frac{\sum_{i,j} p_i^u x_{ij}}{\sum_{i,j} p_i x_{ij}} - (1 + \xi_u) g_u, 0 \right), \max \left((1 - \xi_u) g_u - \frac{\sum_{i,j} p_i^u x_{ij}}{\sum_{i,j} p_i x_{ij}}, 0 \right) \right)$$

$$\Lambda_3(V) = \max \left(\chi - \frac{\sum_{i,j} q_i x_{ij}}{\sum_{i,j} p_i x_{ij}}, 0 \right)$$

$$\Lambda_4(V) = \max \left(\sum_i D_i v_i - \hat{K}, 0 \right) + \max \left(\hat{L} - \sum_i E_i v_i, 0 \right)$$

Specifically, $\Lambda_1(V)$ considers the original objective function. $\Lambda_2(V)$ and $\Lambda_3(V)$ consider the violation of type 1 constraints, while the former accounts for faculty fairness and the latter accounts for student preference. $\Lambda_4(V)$ considers the overall violation of type 2 constraints indirectly by aggregating (4.3) and aggregating (4.4). $\max \left(\sum_i D_i v_i - \hat{K}, 0 \right)$ represents the violation of the room capacity constraint. If it is positive, it means that even if we can find a room assignment such that each room in every timeslot is occupied by one module, we still have some modules that cannot be assigned to. Similarly, $\max \left(\hat{L} - \sum_i E_i v_i, 0 \right)$ represents the violation of the utilization constraint, i.e., the unfulfilled overall utilization according to the target utilization level.

In every step of the construction of approach 1, we first randomly select one objective as the ranking criteria according to the probability

$\{p_n, n=1, \dots, 4\}$. After a criterion n is selected, similar to the idea of the value-based restricted candidate list in GRASP (Delmaire et al. 1999), we compute the performance (known as κ) under that criterion n for all the unselected modules. We then find the κ_{\max} and κ_{\min} which are the maximum and minimum performance values for all the unselected modules. Later, we randomly select a candidate of which the performance is between κ_{\min} and $\kappa_{\min} + \alpha_n(\kappa_{\max} - \kappa_{\min})$, where $\alpha_n \in (0,1)$. This candidate can be viewed as a good candidate for the criterion n , and is added to the selected modules.

Note that p_n represents the preference of criterion n . Initially, p_n is set to $1/4, n=1, \dots, 4$. α_n determines whether the randomness (α_n near 1) or greediness (α_n near 0) is preferred when selecting the next candidate with regard to criterion n . Initially, α_n is set to 0.6, $n=1, \dots, 4$. We may change p_n and α_n from iteration to iteration. First, we compute the performance in term of all four criteria for all the solutions generated so far. Then we identify the criterion n' which is the worst among all four criteria and increase our preference on it, i.e., increase $p_{n'}$ (and decrease preferences on others accordingly). Similarly we increase $\alpha_{n'}$ when we find that the historical solutions evaluated under criteria n'' have smaller variability. Hence, we can explore more solutions under that criterion.

The stopping criteria for the construction are: (1) a solution satisfying type 1 constraints is found; (2) the performance of the solution so far does not

improve in consecutively several steps. In the first case, we obtain a feasible solution for approach 1, while in the second case the reparation is required which is described in Section 4.2.3.

4.2.2 Approach 2: Bi-objective MIP model solved by NBI method

Instead of using the constructive approach which selects module one by one, we build a bi-objective MIP model to find a solution to MSP directly. In this model, the first objective considers the violation of aggregated type 2 constraints. The second objective is a surrogate measure of the inter-campus traffic measure in (3.1). We use this measurement to decrease the complexity of (3.1) when we solve this problem. It uses the information that we learn from the cluster analysis in Chapter 3 to simplify the original objective function by only considering those significant contributors of traffic. All the remaining constraints from P2 are explicitly considered. This bi-objective model is then solved by using the idea of NBI method by Das and Dennis (1996) with modifications by Shukla (2007) which effectively explores the Pareto front. By changing the parameters in NBI method from iteration to iteration when approach 2 is chosen, we can generate various feasible module selections.

The bi-objective MIP model **B-P2** is displayed in the following:

$$[\mathbf{B-P2}] \min \left(\sum_{k=1, \dots, |I|-1} \frac{w_k}{|I_1^k| |I_2^k|} \sum_{l \in I_2^k} F_{kl}, \hat{L} - \sum_i E_i v_i \right) \quad (4.5)$$

Subject to

$$F_{kl} \geq \pm \sum_{i \in I_1^k} r_{il}^* (v_{i_1} - v_l), \forall k = 1, \dots, |I| - 1; l \in I_2^k \quad (4.6)$$

$$F_{kl} \in \mathbb{R}, \forall k = 1, \dots, |I| - 1; l \in I_2^k \quad (4.7)$$

(4.1), (4.2), (4.3)

In this model, the first objective on F_{kl} represents the surrogate inter-campus traffic measurement, in which the set $\{I_1^k, I_2^k, \forall k\}$ represents the information we obtain from cluster analysis. The details will be shown later. The second objective on V is the aggregated measurement about the room capacity constraint (4.4). In general, the smaller this measurement is the higher the possibility that more modules are selected to well utilize the rooms.

The constraints considered in this bi-objective model are type 1 constraints (4.1), (4.2) and the aggregated room capacity constraint (4.3) which is the remaining part of the type 2 constraints. In addition, (4.6) and (4.7) are the auxiliary constraints needed when we linearize the surrogate measure of traffic. Nevertheless, any solution to this model satisfies type 1 constraints, which is different from approach 1 in which violations may occur.

In the following, we first describe how we develop the surrogate measure of traffic. We then describe how to solve this bi-objective problem by using NBI approach.

Surrogate objective function

The motivation of developing the surrogate measure of inter-campus traffic is to reduce the complexity of B-P2 by decreasing the number of auxiliary variables in the model, which are introduced through the linearization of the objective function on traffic. If we use original function (3.1), the cardinality of the auxiliary variables is $\left| \left\{ v_{i_1 i_2}, \forall i_1, i_2 \in I \right\} \right| = O(|I|^2)$. By using the surrogate function, we can show that (in A.2) the cardinality is reduced to $O(|I| \log |I|)$.

The reason can be summarized in the following:

Unlike the original function (3.1) which considers the traffic contribution from all possible campus assignments across all timeslots and directions, our surrogate function only captures those significant ones. By estimating the upper bounds of these captured ones, we penalize the corresponding campus assignments. The significant traffic contributions are identified by using hierarchical clustering method which is linked back to the data analysis work in Chapter 3. In this method, every module is initially considered as an individual group. Then we iteratively identify two groups with significant traffic contribution. The group pair is then used to construct part of the surrogate function. We then combine them into a new group and proceed to the next iteration until no combination can be found, making the surrogate function completely generated.

Specifically, in iteration k , we first obtain the derived traffic contribution $\widehat{F}_{I_m I_n}$ for any group-pair $\langle I_m, I_n \rangle$ by computing

$$\widehat{F}_{I_m I_n} = \max_{t \in T} \left(\sum_{\substack{i_1 \in I_m, i_2 \in I_n \\ i_1 < i_2}} r_{i_1 i_2 t}^1, \sum_{\substack{i_1 \in I_m, i_2 \in I_n \\ i_1 < i_2}} r_{i_1 i_2 t}^2 \right), \text{ i.e., an upper bound of traffic}$$

contribution across any timeslots and directions. Second, among all group-pairs we identify the pair $\langle I_m^*, I_n^* \rangle$ with highest derived traffic contribution such that $m < n$. These two identified groups becomes I_1^k and I_2^k for iteration k . Third, we combine these two groups into a new group to replace the original I_m and proceed to the next iteration. As exactly two groups are combined in one iteration, the process ends in $|I|-1$ iterations. The details of this method are also described in [A.2](#).

The surrogate function is then constructed given $\{I_1^k, I_2^k \mid \forall k\}$. We use

$$F_k(V) = \frac{1}{|I_2^k|} \sum_{i_2 \in I_2^k} \left| \frac{1}{|I_1^k|} \sum_{i_1 \in I_1^k} r_{i_1 i_2}^* (v_{i_1} - v_{i_2}) \right| \text{ where } r_{i_1 i_2}^* = \max_{d \in \{1,2\}, t \in T} r_{i_1 i_2 t}^d \text{ to measure the}$$

traffic contribution of the identified group pair I_1^k and I_2^k . In the worst case scenario when all modules from I_1^k are reallocated, while all modules from I_2^k are not reallocated (or vice versa), the traffic contribution by these modules at

any timeslot and in any direction is bounded by $\sum_{i_2 \in I_2^k} \left| \sum_{i_1 \in I_1^k} r_{i_1 i_2}^* (v_{i_1} - v_{i_2}) \right|$. By

normalizing this measurement and summarizing them with some positive weight to specify the associated penalty, we formulate the surrogate objective

function as $F'(V) = \sum_{k=1, \dots, |I|-1} w_k F_k(V)$. To linearize this surrogate function we

introduce (4.6) and (4.7).

Solving using the idea of NBI

A common way to solve bi-objective optimization problem is to use scalarization technique to explore the Pareto front. The scalarization typically transforms the multi-objective problem into a single-objective problem by introducing additional parameters. This single objective problem is then solved repeatedly by adjusting the value of the parameters so that different subsets of efficient solutions are found (Ehrgott 2006). Considering that the two objectives in our model are in different scales, we choose modified NBI approach, which is thoroughly described by Das and Dennis (1996) and Shukla (2007), as it uses a scalarization scheme and is also independent of the different scales of objective functions.

We briefly describe the process of the method in the following. We also use Figure 4-2 to illustrate. Assuming our bi-objective problem is a minimization problem on S , which is the domain of the decision variable X . The bi-objective vector is denoted as $F(X) = (f_1(X), f_2(X))^T$. Here we assume that $f_1(X)$ and $f_2(X)$ are non-negative³⁸, and the individual optimal values for both are f_1^* and f_2^* which are attained at X_1^* and X_2^* . In the objective space, the two individual optimals are represented by $F(X_1^*)$ (point M) and $F(X_2^*)$ (point N). We find the convex hull (the line segment MN in

³⁸ If not, we can shift the objective function to achieve non-negativity.

our illustrated case) of the individual optimals in the objective space as $conv(F(X_1^*), F(X_2^*)) = \{F(X_1^*)\beta_1 + F(X_2^*)\beta_2 \mid \beta_1 + \beta_2 = 1, \beta_1, \beta_2 \geq 0\}$. We denote the 2x2 matrix $(F(X_1^*), F(X_2^*))$ as Φ and vector $\beta = (\beta_1, \beta_2)^T$. Given a fixed β , we can select a point $X_\beta = \Phi\beta$ on MN (as illustrated in Figure 4-2). The corresponding efficient solution X_β^* can be found by searching for a point within $F(S)$ along the direction (normal to MN and towards the origin, as we solve a minimization problem) such that it is no worse than other points in terms of the two objectives. In other words, we solve the following sub-problem \mathbf{NBI}_β :

$$[\mathbf{NBI}_\beta] \min_{X \in S, t \in \mathbb{R}} t \quad (4.8)$$

Subject to

$$\Phi\beta + t\hat{n} \leq F(X) \quad (4.9)$$

where $\hat{n} = -\Phi(1,1)^T$ is the normal vector pointing from X_β towards the origin of the objective space. By searching for the a larger t , we can find the solution that has better performance in terms of the two objectives. Any optimal solution of \mathbf{NBI}_β can be proved to be weakly efficient (Shukla 2007).

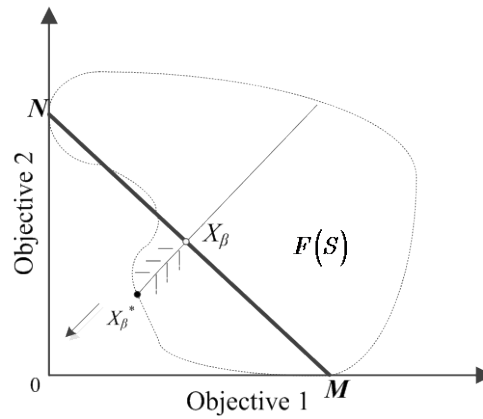


Figure 4-2 Obtaining solutions to a typical bi-objective problem using NBI

The parameter β is typically set as $(n\delta, 1-n\delta)^T$, $n=1, \dots, k$, where δ is the step size and k is the largest integer no larger than $1/\delta$.

We use CPLEX to solve NBI_β (the way of calling CPLEX is described in 3.4). To avoid consuming too much computational time, we set a time limit. If the solver can find an integral solution in time, it becomes the output of approach 2. We then proceed to local-improvement stage. Otherwise, we stop and restart the overall algorithm.

4.2.3 Reparation Mechanism and Local Improvement

In case that the output from approach 1 does not completely satisfy type 1 constraints, we try to repair the infeasibility. We use local search method to explore the neighborhood of the input solution V^\diamond . The neighborhood is defined by 1-exchange, i.e., replace a selected module which contributes to the infeasibility with a module currently unselected. We use beam search method

(Norvig 1992; Zhou and Hansen 2005) to explore the search tree specified by the neighborhood.

Specifically, we initially generate the neighbors of V^\diamond accordingly as the successor nodes of V^\diamond in the search tree. From then on, beam search explores one layer of the search tree in one iteration. It first sorts nodes in the current layer according to a ranking criterion which evaluates the violations of type 1 constraints. Only the best L nodes are chosen to be expanded for the next layer. The stopping criteria are: (1) a feasible solution to MSP is found. (2) no solutions which reduce the violations of type 1 constraints can be found in consecutively several iterations. (3) the objective value is worse than the one of V^\diamond by $\iota\%$ for all the L nodes chosen in one layer, where ι is a threshold.

If we can find a feasible solution to MSP through reparation, the local-improvement method is then applied. Otherwise, the overall algorithm stops and we restart to the next iteration of phase 1. On the other hand, we also apply the local improvement method on the output from approach 2.

The local improvement method tries to find a solution to MSP with a better objective value among those neighboring module selections to the current solution V^\dagger . The neighborhood is defined by 1-move (add/remove one module) followed by 2-move (add/remove two modules). Specifically, we first identify the timeslot and direction such that the traffic component is the highest for V^\dagger . Only applying changes on the module selection related to the identified timeslot and direction can help improve the objective value. We try

to add/remove modules accordingly to reduce the objective value. If a feasible solution can be found, it becomes the output of the local improvement phase. Otherwise, we additionally check all possible 2-moves. Similarly, the best feasible neighbor improving the traffic value becomes the output if it exists. Otherwise, we check those infeasible ones such that they can improve the objective value while the violations to type 1 constraints are not severe. If any of these infeasible solutions can be repaired using the reparation process we just described, it also becomes the output of the local improvement phase. If not, the local improvement cannot find any better solution.

4.3 Phase 2: Room Assignment Problem

The goal of the second phase is to generate a feasible room assignment solution based on the output of phase 1. The main task is to search for a feasible room assignment for those modules selected. If we can find at least one, the assembled complete solution is also feasible to [P1](#). In the case of failure, we restart to the next iteration.

For the room assignment problem, we do not need to use two sets of variables $\{x_{ij}\}$ and $\{y_{ij}\}$ to separately model room-assignment-decisions for lectures and tutorials. This is because constraint (3.9) in [P1](#) requires that if a module is reallocated, its lectures and tutorials are reallocated. As a result, we introduce a set \hat{I} which contains the indices of the lectures and tutorials of all selected modules. For convenience we still call each element in \hat{I} “module” (although it can be either lecture or tutorial) in phase 2, and the decision

variable in room assignment problem can be denoted by $\{\hat{x}_{ij}, \forall i \in \hat{I}, j \in J\}$.

The room assignment problem, which is to search for a feasible room assignment given the corresponding module selection, is modelled by the following satisfaction problem **P3** on $\{\hat{x}_{ij}\}$:

[P3] Find a $\hat{X} = \{\hat{x}_{ij} \in \{0,1\}, \forall i \in \hat{I}, j \in J\}$

Subject to

$$\sum_i d_{it} \hat{x}_{ij} \leq k_j, \forall t \in T, j \in J \quad (4.10)$$

$$\sum_i -e_i \hat{x}_{ij} \leq -l_j, \forall j \in J \quad (4.11)$$

$$\sum_j \hat{x}_{ij} = 1, \forall i \in \hat{I} \quad (4.12)$$

Constraints (4.10) and (4.11) are modified type 2 constraints (3.11) and (3.12), and we introduce d_{it} and e_i to replace original parameters d_{it}^1 , d_{it}^2 , e_i^1 and e_i^2 . Constraints (4.12) ensure that all of the reallocated modules must be assigned to some room type. If we can find a feasible solution to **P3**, we obtain the complete feasible solution by mapping from \hat{I} back to I .

Solving a feasibility integer programming problem, or proving the infeasibility, is very complicating. One way to solve feasibility problem is to transform it into an optimization problem. Therefore, we transfer **P3** to the following optimization problem **P4** by relaxing (4.12):

$$[\mathbf{P4}] \quad \max_{\hat{x}_{ij} \in \{0,1\}} \sum_{i,j} \alpha_i \hat{x}_{ij} \quad (4.13)$$

Subject to

(4.10), (4.11)

$$\sum_j \hat{x}_{ij} \leq 1, \forall i \in \hat{I} \quad (4.14)$$

In P4, (4.12) is replaced with (4.14). We call (4.14) the disjointing constraints. It relaxes (4.12) as it allows that a module can be assigned to no rooms at all. In the objective function, we introduce a weight coefficient α_i which measures the importance of a module.

If there is an optimal solution to P4 and the objective value equals $\sum_i \alpha_i$, the solution is also a feasible solution to P3. It is because this solution satisfies (4.10), (4.11), while $\sum_j \hat{x}_{ij} = 1$ for any \hat{x}_{ij} . In this way, instead of solving P3, we solve the equivalent MIP model P4.

4.3.1 Overall Framework

We use branch-and-bound approach to solve P4 progressively. We branch on the possible room assignments for the selected module. We use Lagrangian relaxation method to decompose the sub-problem into a collection of Multidimensional Knapsack Problems (MKPs) in order to find the dual bound, and use a primal heuristic to find a feasible solution in case that the result from the Lagrangian relaxation method is not integral. Various studies have

reported to incorporate Lagrangian relaxation into the branch-and-bound framework (Fisher 1981; Holmberg and Yuan 2000).

We branch on all possible room assignments for module $i' \in I$. For instance, if module i^\diamond is the next branching module, and it can be assigned to room types j_1 and j_2 , we generate two branches. One branch is $x_{i^\diamond j_1} = 1$ with $x_{i^\diamond j'} = 0, \forall j' \neq j_1$, and the other branch is $x_{i^\diamond j_2} = 1$ with $x_{i^\diamond j'} = 0, \forall j' \neq j_2$.

We use the typical cut-off criteria in branch-and-bound framework to help either prune sub-tree or serve as the stopping criteria. Specifically, the sub-tree is cut (1) if the feasible solution newly found assigns all items; (2) or if no feasible solution is found by executing the primal heuristic; (3) or if dual bound is worse than the best primal bound.

To select the next branching module, we use the information learned from the result of solving the sub-problem in the branch and bound framework. The module that has the highest number of room types to be assigned becomes the next branching module (This is because the relaxation made in the sub-problem allows that one module can be assigned to more than one room type). The generated branches are sorted by the room utilization. Specifically, we compute the ratio of the current utilization level to the target level set in utilization constraint (4.11) for the room types associated to all branches. The branch that has the lowest ratio is sorted to the first place.

In general, we use depth-first-strategy to explore all the branches to the leaf layer. The general flowchart of this branch-and-bound framework is

shown in Figure 4-3. We then describe how to obtain dual and primal bound for each node in the branching tree.

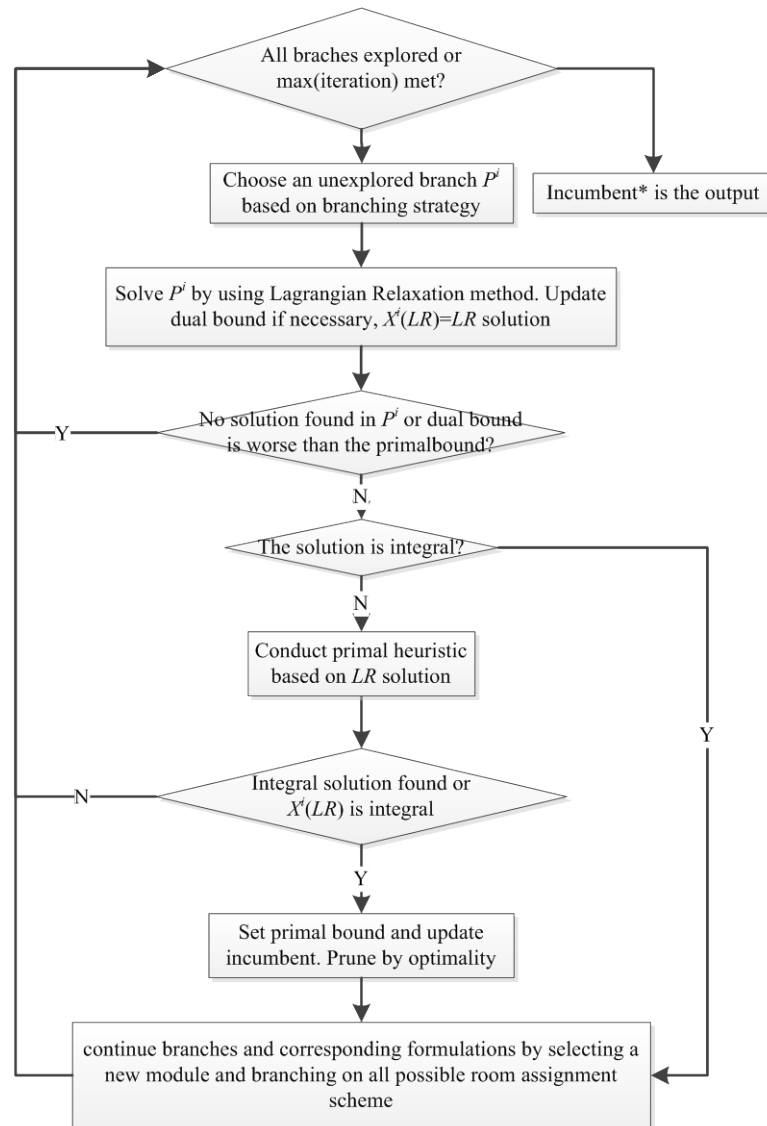


Figure 4-3 Flow chart of the branch-and-bound framework in Phase 2

4.3.2 Dual Bound: Lagrangian Relaxation Method

Solving **P4** directly seems to be challenging, especially when $|\widehat{I}|$ is large. The complication includes not only the huge scale of decision variables, but also a variety of complicating constraints that restrict us to making use of several potential problem structures. In our proposed method, we use Lagrangian method to relax constraints (4.11) and (4.14), i.e., both demand and disjointing constraints. The relaxed problem **LR** is shown in the following:

$$[\mathbf{LR}] \max \sum_{i,j} \tilde{c}_{ij} \widehat{x}_{ij} + \sum_i \mu_i - \sum_j \nu_j l_j$$

Subject to

$$\sum_{i \in I} d_{it} \widehat{x}_{ij} \leq k_j, \forall t \in T, j \in J$$

where $\tilde{c}_{ij} = \alpha_i - \mu_i + e_i \nu_j$. The associated dual variables are

$$W = \left\{ \left\{ \mu_i \right\}, \left\{ \nu_j \right\} \right\}.$$

To solve the Lagrangian dual of **LR**, we use sub-gradient method. It is used to guide the search for better Lagrangian multipliers to improve the Lagrangian function value iteratively. With initial Lagrangian multipliers arbitrarily set, we can compute the sub-gradient from the dual variables' values in any specific iteration. With the sub-gradient computed which contributes the determination of the search direction, we in turn update the dual variables for the next iteration. We briefly describe the way to use the sub-gradient to update the dual variables W in Algorithm 4-1.

Specifically, the way that we obtain the step size follows the one described in various studies (Polyak 1967, Sen and Sherali 1986 and Wolsey 1998). The way that we set the search direction follows the one described by Gaivoronski (1988). Maximum iteration k^* varies according to the current layer of the sub-problem. We give more allowable iterations when we explore deeper into the tree.

Algorithm 4-1: Sub-gradient method to solve LR**input** MIP model LR.

- 1 Initialize the Lagrangian dual $W^{(0)}$: Set $\{\mu_i\}$ to zero. Set $\{\nu_j\}$ to the dual prices reflected by the solution to the following LP: $\max_{\hat{x}_{ij} \in \{0,1\}} \sum_{i,j} \alpha_i \hat{x}_{ij}$ s.t. (4.11). Set count $k=0$.
 - 2 **repeat**
 - 3 Solve LR given $W^{(k)}$. Denote the solution as $\{\hat{x}_{ij}^{*(k)}\}$ and objective value as $\bar{z}^{(k)}$.
 - 4 Set $\Delta^{(k)}$ by $\Delta_i^{(k)} = 1 - \sum_j \hat{x}_{ij}^{*(k)}$ for each μ_i and $\Delta_j^{(k)} = \sum_{i \in I} e_i \hat{x}_{ij}^{*(k)} - l_j$ for each ν_j .
 - 5 Set step size $s^{(k)} = \lambda_k \frac{\bar{z}^{(k)} - \underline{z}^{(k)}}{\|\Delta^{(k)}\|^2}$, where $\underline{z}^{(k)}$ is an upper bound of LR.
 - 6 **if** $k = 1$ **then**
 - 7 Set direction $d^{(k)} = \Delta^{(k)}$.
 - 8 **else**
 - 9 Set direction $d^{(k)} = (\Delta^{(k)} + \theta d^{(k-1)}) / (1 + \theta)$, $\theta \in (0,1)$.
 - 10 **end if**
 - 11 Update dual variables $W^{(k+1)} = W^{(k)} + s^{(k)} d^{(k)}$. $k = k + 1$.
 - 12 **until** $k = k^*$
- output** best feasible solution obtained so far.

We do not solve LR directly. Since this problem is separable, we can decompose LR into $|J|$ (the number of room types) multidimensional knapsack problems. For a specific room type $j' \in J$, we solve the following sub-problem **MKP** $_{j'}$:

$$[\text{MKP}_{j'}] \max_{\hat{x}_{ij'} \in \{0,1\}} \sum_i \tilde{c}_{ij'} \hat{x}_{ij'}$$

Subject to

$$\sum_i d_{it} \hat{x}_{ij'} \leq k_{j'}, \forall t \in T \quad (4.15)$$

The sub-problem is a MKP³⁹ and there are at most $|T|$ (the number of timeslots) dimensions. This problem tries to pack a set of items (in our case, modules) to a special knapsack (in our case, room type j'), which has more than one dimension. Accordingly, each item may have different weights for all dimensions, and a feasible packing of items must not exceed the capacity of all dimensions. The objective is to gain the most profit from the packing. The constraints (4.15) are usually referred as knapsack constraints.

Solving a MKP in practice is still non-trivial as it is a strong NP-hard combinatorial optimization problem (Puchinger, Raidl, and Pferschy 2010). One can solve it by using the exact approach, approximation approach or meta-heuristics approach. Exact solution techniques include branch-and-bound and dynamic programming (Gottlieb 2000). Both approaches need considerable time to find the optimal of large scale problem. As for the approximation technique, even when there are only two dimensions, there exists no Fully Polynomial-Time Approximation Scheme (FPTAS) unless $P=NP$. Typical PTAS such as a ε -approximation H^ε -ext-greedy algorithm

³⁹ As a preprocessing, we can always set those $x_{ij'}$ to 0 such that $\tilde{c}_{ij'} \leq 0$, since $d_{it} \geq 0, \forall i, t$.

has a running time of $O\left(n^{\lceil d/\varepsilon \rceil - d}\right)$, where n is the number of items and d is the number of dimensions (Caprara et al. 2000). Apparently when the problem gets bigger, a PTAS is not good enough. Since the number of items and dimensions are relatively large in our case, we try to develop a heuristic algorithm which can yield a good feasible solution given a relatively reasonable period of time.

To solve each MKP_j , using a heuristic, we first try to reduce the number of variables. We adopt the idea of the core concept in the context of knapsack problems (Kellerer, Pferschy, and Pisinger 2004). The idea is that among all the modules to be assigned, those with very high ‘efficiency’ is almost certain to be assigned, while those with very low ‘efficiency’ is almost certain not to be assigned. The remaining ones, known as the ‘core’⁴⁰, are hard to decide and hence left for further investigation. Core is usually solved by heuristics such as GA. General discussion on how to adopt core algorithm in MKP is described by Puchinger, Raidl, and Pferschy (2010). The following three paragraphs discuss that (1) what is the definition of ‘efficiency’; (2) After sorting modules by efficiency, how to conduct the variable fixing; (3) After the variable fixing, how to solve the reduced problem.

⁴⁰ Technically the core is defined in slight different way in studies. However, for simplicity we use the term to describe the main idea.

The efficiency of module i is defined as $\hat{e}_i = \frac{\tilde{c}_{ij'}}{\sum_t d_{it} \mu_t}$ where μ_t is the

dual variables associated with the knapsack constraint of dimension t (Hence, μ_t is obtained by solving the Linear Programming (LP) relaxation of MKP $_{j'}$).

Sorting by \hat{e}_i for all modules helps us to identify three ordered groups of

modules: Groups $\widehat{I}_\alpha, \widehat{I}_\beta, \widehat{I}_\gamma$. They represent the collection of modules of which

the efficiencies are higher than one, equal to one and lower than one

respectively. However, we find that for those modules in \widehat{I}_β , the

corresponding solutions to the LP relaxation can be either fractional or integral,

and usually $|\widehat{I}_\beta|$ is large. Therefore, we group the modules in \widehat{I}_β into three

sub-groups. \widehat{I}_{β_1} contains those modules of which the LP solution equals 1,

\widehat{I}_{β_2} contains those modules having fractional solution, while \widehat{I}_{β_3} contains

those modules of which the LP solution equals 0. We introduce another

efficiency to further sort these two sub-groups: $\hat{e}'_i = \frac{\tilde{c}_{ij'}}{\sum_t d_{it} \left(\sum_i d_{it} - k_{j'} \right)}$.

The core is generated based on the sorted modules obtained in the last

paragraph. Starting from the \widehat{I}_{β_2} , we obtain the core by expanding toward two

directions on the sorted list until the expanded list reaches a limit of size.⁴¹ After that, in the sorted list the modules before the core are fixed to be assigned to room type j' , while the modules after the core are fixed not to be assigned to j' . By using this variable fixing, the original $MKP_{j'}$ is reduced.

To solve this reduced problem, we use GA as it has been successfully applied to MKP in various studies (Gottlieb 2000, Raidl and Gottlieb 2005). We use the direct representation as the genetic representation, which is discussed thoroughly by Chu and Beasley (1998). We choose uniform crossover and bitwise mutation as the genetic operators, which are typical operators for our genetic representation. We use the objective function of $MKP_{j'}$ as the fitness function. The population size is set to $\lfloor \{d\} / 2 \rfloor$. We choose fitness proportionate selection strategy (i.e., we filter out any explored solutions) as the selection strategy which prefers those candidates with better objective value (Back, Fogel, and Michalewicz 1997). To construct the initial population, we solve the LP relaxation of the reduced problem and compute the efficiency \hat{e}_i for all modules followed by sorting modules by \hat{e}_i . As the sorted list can be viewed as a permutation, we pack modules one by one until any knapsack constraints (4.15) are violated. In case we obtain an infeasible

⁴¹ Through experiments we found that any integer between $0.15|i|$ and $0.20|i|$ shows the best performance when we comparing the result from solving core problem with the result from solving the original problem using solver.

solution during genetic operations, we use a simple reparation way which is to remove items from the pack based on the sorting by \hat{e}_i until the remaining of the packing becomes feasible.

4.3.3 Primal Bound: Constraint Programming-Based Heuristic

The primal bound is important in phase 2 because (1) it helps to find more primal bounds to prune the branching tree; (2) it helps the sub-gradient search to find better step size and hence improve its convergence. If Lagrangian method can find a feasible solution, a primal bound is found. However, this may not happen very often. If the solution, denoted by $\hat{X}^L = \{x_{ij}^L\}$, is not integral, we use the following method to obtain a primal solution by repairing infeasibility of \hat{X}^L . Within the reparation, we solve the following feasibility problem **P5**:

[P5] Find a $\hat{X} = \{\hat{x}_{ij} \in \{0,1\}, \forall i \in \hat{I}, j \in J\}$

Subject to

$$\sum_i d_{it} \hat{x}_{ij} \leq k_j, \forall t \in T, j \in J \quad (4.16)$$

$$\sum_i e_{ij} \hat{x}_{ij} \geq l_j, \forall j \in J \quad (4.17)$$

$$\sum_j \hat{x}_{ij} = 1, \forall i \in \hat{I} \quad (4.18)$$

$$\hat{x}_{ij} = x_{ij}^L, \forall \langle i, j \in \bar{\mathcal{O}} \rangle$$

P5 is similar to P3 except two points. First, decision variables are additionally fixed based on \widehat{x}_{ij}^L . \mathfrak{U} contains those module-room type combinations $\langle i, j \rangle$ such that \widehat{x}_{ij}^L is (1) integral, (2) $\sum_j \widehat{x}_{ij}^L \neq 0$ (3) $\sum_j \widehat{x}_{ij}^L \leq 1$. In other words, integral variables are fixed as long as they does not instantly create infeasibility in P5. Second, we replace the original parameter e_i with parameter e_{ij} . $e_{ij} = e_i$ if $j \in \widetilde{J}_i$ and $= 0$ otherwise. Its usage is described later.

We use CP technique to solve P5. The variables in the CP model are $\{\widehat{x}_{ij} \mid i, j \notin \mathfrak{U}\}$ (variables not fixed by \widehat{X}^L). The domain of the CP model is initiated from constraints (4.18) accordingly. The constraints of the CP model are hence (4.16) (4.17) with the following derived new constraints:

$$\sum_i e_{ij'} - l_{j'} \geq \sum_i e_{ij'} \sum_{j \neq j'} x_{ij} \geq \sum_{j \neq j'} l_j + \sum_{\substack{i \in I_{j,j'}^\alpha \\ j \neq j'}} e_i x_{ij} - \sum_{\substack{i \in I_{j,j'}^\beta \\ j \neq j'}} e_i x_{ij}, \forall j' \in J \quad (4.19)$$

where $I_{j_1, j_2}^\alpha = \{i \mid e_{ij_2} = 0 \wedge e_{ij_1} = e_i\}$ and $I_{j_1, j_2}^\beta = \{i \mid e_{ij_2} = e_i \wedge e_{ij_1} = 0\}$.

Constraint (4.19) uses the fact that for an item $i' \in \widetilde{I}$, the set $\{e_{i',j}\}$ only contains two possible values: $e_{i'}$ if i' can be assigned to room type j and 0 otherwise.

The CP model is solved by CPLEX CP solver. If we can find a feasible solution for P5, we obtain a primal solution.

4.4 Numerical Experiments

Before we show the numerical experiments and results, we first describe the test cases we use throughout this section. In reality the parameter settings for the MRPT may vary significantly from university to university. The data we used in previous chapter only covers one university and several academic years, so it could bring bias. In addition, different parameter settings may also affect the “difficulty” of the problem as well as the performance of the solution technique. For these reasons, we provide a way to generate test cases with various parameters based on what we have learned from the stakeholders’ perspective. In [A.1](#), we provide the details on how to generate such test cases. In general, we first want to represent two typical problem scales which reflect medium and large-sized universities. Second, we want to use different constraint settings to represent various resource scarcities, preference on the fairness in terms of sharing from faculties and student level. For the second part, the settings are adopted by those scenarios set by the stakeholders.

Different test cases have different groups of students, module registration information, different timetables, etc., but they share some similarities so that we can compare performance on different scenarios. Particularly, these cases can be first categorized into two groups. One is for cases such that 400 modules (medium-sized university) are considered and the other is for cases such that 800 modules (large-sized university) are considered. Each test groups contain 100 random generated test cases. Each test case is associated with three different constraint settings to form three different test

instances. These three settings represent loose, medium and tight constraints and we call them low, medium and high settings. They are obtained by communicating with stakeholders and being carefully selected from many candidate scenarios as they are believed to be representative. Hence, we use the combination of the number of modules considered and the constraint tightness settings to form a test case. For instance, 400 low refers to a test case such that 400 modules are considered and the constraint is loose.

The test environment is dual-core PCs running Windows 7 64bit OS on 5GB of memory. We use multi-core system so that the commercial solver should benefit from its parallel computing features. Choosing 64bit OS and larger-than-4GB-memory should ensure that system memory cannot be a bottleneck for commercial solver. Once again CPLEX 11 is used as the commercial MIP solver. We set the 'MIPEmphasis' to 'automatic', and the solver stops at exact optimal. All complete feasible solutions which are found by our heuristics and CPLEX are managed to be recorded.

4.4.1 Numerical experiments related to Phase 1

Three numerical experiments related to the proposed phase 1 were conducted. First, we want to show the effectiveness of our alternating two approaches (described in Section 4.2.2). We compare the performance using our proposed way of combining approach 1 and approach 2 with performances using other simpler ways. Second, we show the correlation between our proposed surrogate objective function (described in Section 4.2.24.3.2) and the original objective function. Third, we show that the local improvement method

(described in Section 4.2.3) is effective. We compare the performances between using local search method in phase 1 and not using it.

Experiments on alternating two approaches

Recall that we propose two approaches to solve the phase 1 problem, i.e., MSP. We also propose a way to allow both approach 1 and approach 2 to be run for several iterations in phase 1, and the approach which achieves a better overall historical performance has a higher chance to be selected. In this experiment, we show that combining two approaches yield better result than using only one of them.

Particularly, we compare the following four different scenarios.

- (1) Approach 1 only: we only use approach 1 through phase 1 iteration. Reparation (if necessary) and local improvement are conducted after approach 1 is finished.
- (2) Approach 2 only: we only use approach 2 through phase 1 iteration. Local improvement is conducted after approach 1 is finished.
- (3) Randomly select approach 1 and approach 2: We use both approaches in phase 1 iteration. However, the selection of approach is purely random. According to which approach is selected, reparation (if necessary) and local improvement are conducted accordingly.

- (4) Our proposed way.

We run the heuristic on our test cases. For each scenario, we counted the number of cases in which it can find the strictly best solution among all scenarios. We used two sets of experiments with different time budget to test the short-term and long-term performances. For the short-term performance experiment, the time budget was 1 hour time. For the long-term performance experiment, the time budget was 10-hour.

Table 4-2 The count of wins for four scenarios given 1-hour computational budget

Scenarios	I =400			I =800		
	Low	Medium	High	Low	Medium	High
1	1	2	0	1	0	0
2	8	10	33	20	25	37
3	5	3	6	3	4	2
4	84	80	60	74	65	52

From the results shown in Table 4-2, we find that our proposed method had comparatively the best results. Scenario 1 yielded the worst result. In addition, the performance of scenario 2 was much better than scenario 1 and 3. In some cases, it even achieved relatively good performance comparing to scenario 4 (For instance, 800-high). These results can be explained by the following reason: Scenario 4 was based on overall historical results, so it may need some learning time. Moreover, the more difficult the test case were, the better results scenario 2 outputed. This can be explained by the fact that the quality of the output of approach 2 was better than approach 1 in general. Specifically, the type 1 constraints were explicitly considered in approach 2.

In addition, other scenarios were considered by solving a multi-objective model rather than constructing by randomly selecting criterion. As a result, using approach 2 can obtain better solutions when compare to using approach 1 only. For scenario 3, as the selection of approach is purely random, the results from iteration to iteration can be too diverse. If approach 1 was primarily selected, the performance can be even worse than scenario 2, which is shown in Table 4-2.

The superiority of scenario 4 was also observed in the long term. From the results shown in Table 4-3, we observe that the all other scenarios were only able to outperform scenario 4 in very limited cases.

Table 4-3 The count of wins for four scenarios given 10-hour computational budget

Scenarios	I =400			I =800		
	Low	Medium	High	Low	Medium	High
1	0	0	0	0	0	0
2	1	0	0	1	1	0
3	1	1	0	0	1	0
4	96	98	100	95	94	97

The frequency (how many times does one approach is called) of approach 1 and approach 2 were also examined. In this case, we used the overall framework to solve individual test cases and record the number of calls of each approach. We summarized the average frequency of approach 1 in Table 4-4. It can be found that approach 2 is tend to be selected more frequently in the more difficult cases, which supports the design intention mentioned in 4.2.

Table 4-4 Frequency of approach 1 and 2 being called when solving overall problem

Scenarios	I =400			I =800		
	Low	Medium	High	Low	Medium	High
Approach 1 frequency (%)	65%	62%	53%	64%	57%	49%

Correlation between surrogate objective function and the original one

Ideally, the surrogate measure of inter-campus traffic should be positively correlated with the original measure. We used the Person correlation coefficient for samples of both $F'(V)$ in B-P2 and $F(x)$ in P1 (described in Section 4.2.2 and 3.3 respectively). In terms of the sample, we needed to enumerate all the possible objective values to compare the correlation. As a result, all possible module reallocation decisions were enumerated, no matter they were feasible to MRPT or not. Due to the limitation of computation resources, when the scale of the test case became bigger, we could only do this implicitly, i.e., randomly generated a subset of solutions. Nevertheless, after we calculated objective values for different test case, we obtained the corresponding correlation coefficient. Note that to simplify the process, we let $w_k = 1, \forall k$ in (4.5).

As a simple presentation, we first generated a test case with only 10 modules. As the size of domain in this case could be handled, we sampled all the 2^{10-1} solutions. In order to illustrate the correlation of two measurements, we plotted the values of F' (surrogate) against F (original) in Figure 4-4. In

that figure, X-axis represents all the $F(x)$ values, and the Y-axis represents the corresponding F' value. The Pearson's correlation coefficient for this specific case was 0.661 so the two are positively linear correlated. In addition, for those solutions such that $F = 0$, the corresponding $F' = 0$ as well, and vice versa. It should be noted that due to the small amount of modules in this case, traffic parameters had many zeroes, and so the largest function value of objective function in this case was much less than the one in larger-scale test case which can easily go up to hundreds.

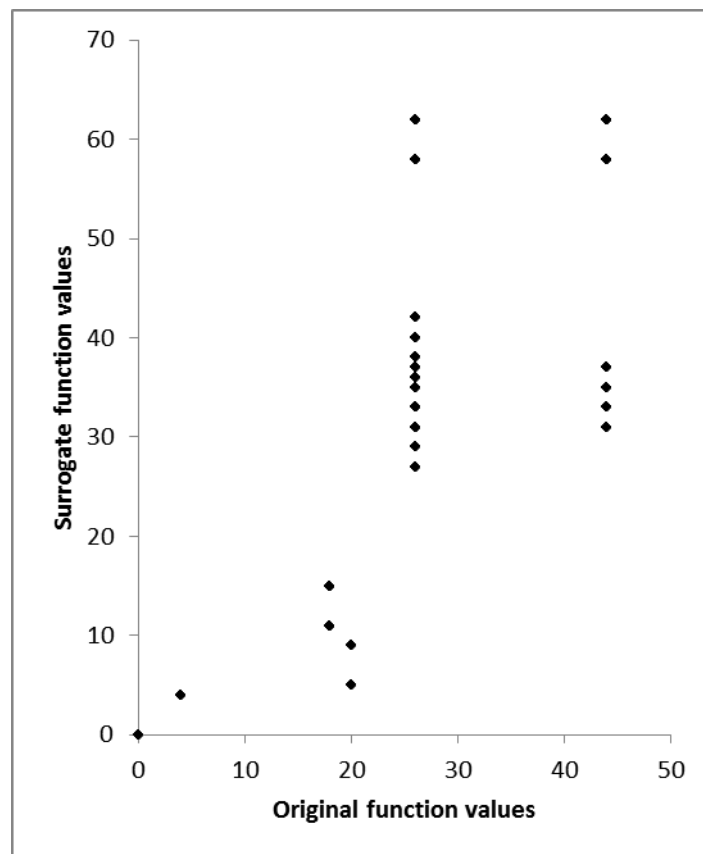


Figure 4-4 Plot of the set of $(F(x), F'(x))$ from a 10-module example

Table 4-5 Average Pearson's Correlation Coefficient (PCC) between $F(x)$ and $F'(x)$ with different numbers of modules

	 I =20	 I =100	 I =200	 I =400	 I =800
Avg. PCC	0.774	0.801	0.839	0.872	0.861

We conducted more experiments by increasing the scale of test cases. We used five groups of test cases with different domain sizes. Each test group contained 100 test cases. In the first group, 20 modules were included in each case, and we still sampled the whole domain (including 2^{19} solutions). In the other four groups, 100, 200, 400, 800 modules were included respectively. As the domain became too big in these cases, we randomly generated 2^{20} distinct solutions for each test case. Table 4-5 shows that all cases show positive correlations and the average coefficients across groups are all over 0.8. Moreover, it seems that there was no trend that the correlation coefficient decreases when the number of modules increases. As 800-module is a practical representation of the scale of a large university, it can be concluded that the surrogate measure is positively correlated to the original traffic measure, i.e., is a representative surrogate measure.

Experiments on local improvement

We conducted experiments to show the percentage of improvement in performance when comparing between not conducting local improvement and conducting it. The difference was shown in terms of percentage of improvement. The computation budget is 24-hour.

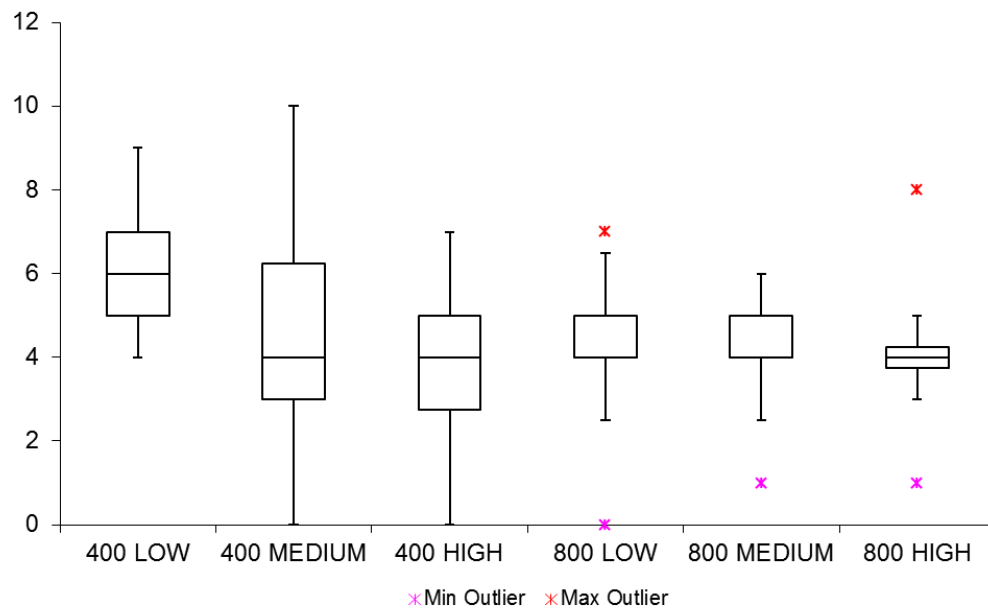


Figure 4-5 % improvement from enabling local improvement

Figure 4-5 shows that enabling the local improvement is able to bring promising improvement. Particularly, for $|I|=400$ cases the improvement was about 4-6%, or about 10-20 in terms of value; for $|I|=800$ cases the improvement was about 4-5%. In terms of inter-campus traffic measurement, the improvement was about 15-25. In addition, we observed several cases (16 out of 600) that no improvement was made. However, for these cases, the time of obtaining the best solution had been decreased (9% on average), indicating that small improvement is still achieved.

4.4.2 Numerical experiments related to Phase 2

We compared the performance of solving P4 between our phase 2 method and the commercial solver. We first tracked the solution progress of our heuristic. We then separately used CPLEX to solve P4 by allowing the same amount of time as our heuristic uses. The way we used CPLEX to solve a satisfaction

problem is imposing an empty objective function. We listed the average rate that CPLEX cannot find a feasible solution quicker than our heuristic over the total number of phase-2 executions for each category we used so far.

Table 4-6 The rate comparison in solving phase 2 problem between our heuristic and CPLEX

Rate representing that our heuristic outperforms CPLEX					
 I =400			 I =800		
Low	Medium	High	Low	Medium	High
94.7%	98.9%	99.9%	99.2%	99.6%	99.9%

The results showed that our proposed phase 2 approach, which used the branch-and-bound framework, outperformed the CPLEX in most cases. Particularly, when the problem becomes difficult, we observed the rate was nearly 100%. The possible reason may be that our proposed heuristic exploited the good structure of P4 which cannot be identified by CPLEX.

4.4.3 Results of the Proposed Heuristic

In this section we compared the results between our proposed iterative two-phase heuristic and CPLEX. The intention is to show the overall performance gain from our proposed method. We compared the performance by presenting the objective value of the best-so-far solution by given same computation time. The time budget was 24 hours for all test cases.

Before we describe the main findings, we should highlight that the solver does not stop in any test case during our experiments, which means that no single optimal solutions can be found given the time budget. The closest-

to-optimal solution was obtained with 0.79% gap (best-so-far: 329.74) in a test case of low-400.

The results of the experiments in this section are described in the following order: We first show the performance comparison between our heuristic and the solver on one single test case. The comparison is illustrated by showing the inter-campus traffic measurement of the best-so-far solutions obtained by each method over time. After demonstrating the result on only one test case, we show the results over all test cases. We begin from showing the improvement of objective value gained by our proposed heuristic from a high level perspective. We use box chart to show the range of improvements across different sets of test cases. Then, we present the win/loss result over time by counting how many times that our heuristic outperforms the solver and vice versa after some period of time.

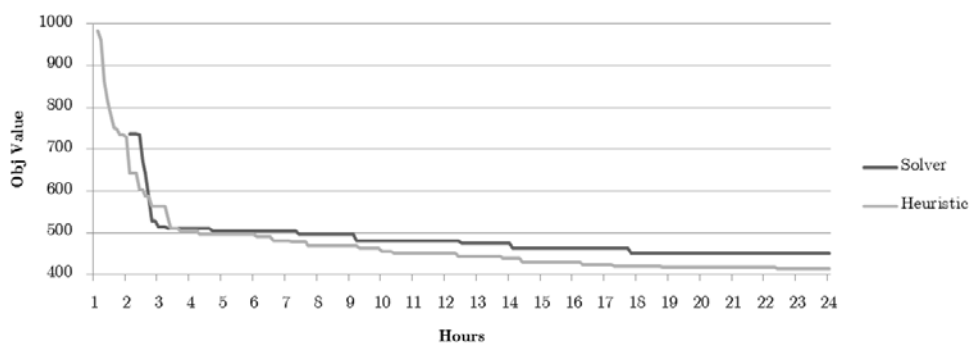


Figure 4-6 Example: performance comparison over time on a 800-medium test case

We start from illustrating the example test case. In Figure 4-6, we present the solution-quality-over-time of our proposed heuristic and the commercial solver. The test case was an 800 medium one. From this particular result we can obtain several observations. First, the solver tended to find the

feasible solution slowly. In this specific face, it took nearly one hour to obtain its first feasible solution, while the heuristic almost found a feasible one instantly. However, the quality of the first feasible solution obtained by the solver was much better than the first one from the heuristic, while it was still worse than the best-so-far heuristic solution at the same time. Second, during the 3rd hour, the solver outperformed our heuristic for about half an hour. However, after that solution quality was constantly inferior to the one of our heuristic. The difference between both became bigger and bigger. At the end the best solution obtained by heuristic was about 40 less than the one obtained by solver. This amount of a traffic rate equals more than half of the full load of a typical shuttle bus. Hence, it is a significant difference.

Then, we show the improvement in terms of original inter-campus traffic measurement gained by our heuristic. Figure 4-7 shows that on average the heuristic can obtain from 20-40 improvement roughly, although there are several cases that the solver beats our heuristic. In particular, $|I|=800$ test cases gained more improvement than $|I|=400$ cases. The main reason was that more modules were included so that the objective value should be larger, and the space for improvement in finding better solution may be larger as well.

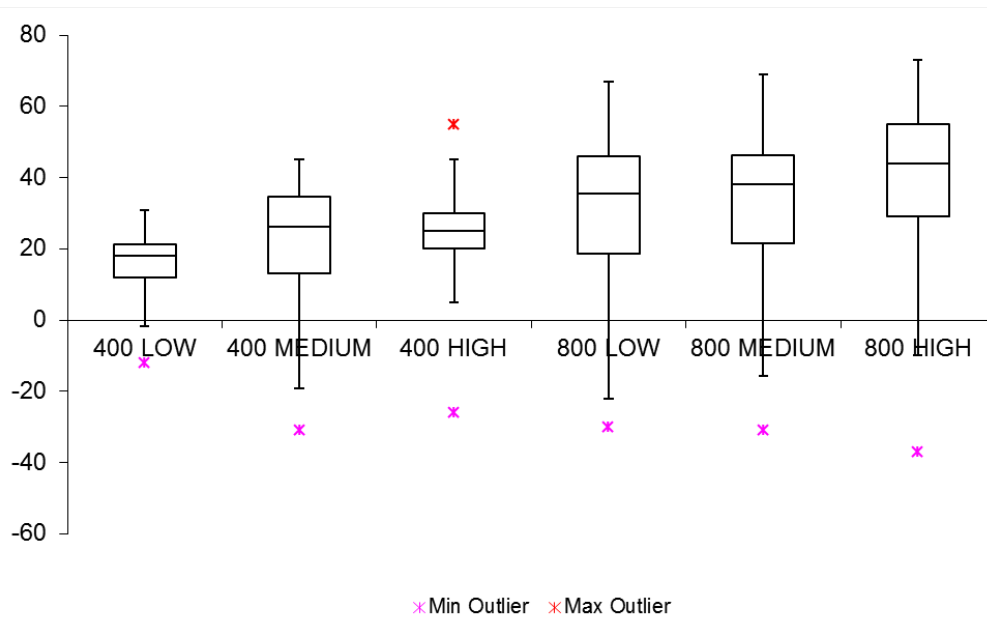


Figure 4-7 The traffic measurement comparison

Lastly, because technically both approaches did not stop during this time, we counted the number of cases that our heuristic beats solver (means the best solution is strictly better. It is denoted as N^+) and the number of cases that our heuristic was beaten (denoted as N^-) in every 1 hour. Therefore, row t represents the values of N^+ and N^- given t hours of running both methods on 600 test cases respectively, including combinations of three constraint settings and two problem scales. If one approach terminated unexpected (such as runs out of memory), the best so far solution stays in the remaining hours.

Table 4-7 shows that in general our heuristic beats the solver in all test settings. The average N^+ / N^- ratio was 20.8, and average $(100 - N^-) / N^-$ ratio was 23.9. This roughly means that, on average, the solve could only beat our heuristics at most once in every 20 tries. In the worst case, $\min N^+$ was 63, while $\max N^-$ was 17, and both happened in the very early stage of the

computation. Comparably, in the end of the process (23 and 24 hours), N^+ was at least 84, while N^- was at most 8. We analyzed the N^+ / N^- ratio across hours, and we found that the ration across the first 4 hours was relatively low (about 7% on average) but increased very fast in later stage. It indicates that when the time budget was not extremely limit, our heuristic outperformed the solver in general.

To give a clearer demonstration, we also provide the line charts for both $|I|=400$ and $|I|=800$ cases to discover trends in Figure 4-8 and Figure 4-9. Both charts show that in general the N^+ increased when more computational hours are allowed, while N^- decreased accordingly. Moreover, it seems that the tighter the constraints were, the smaller N^+ was. However, similar trend was not significant for N^- . This observation indicates that the more difficult the problem was, the more difficult a better solution could be found. Another trend was that when comparing results from $|I|=400$ and $|I|=800$ case, the three counts under low, medium and high settings tended to converge in the latter case. It indicates that when the problem scale became large, the performances of both methods became less sensitive to the tightness of the constraint.

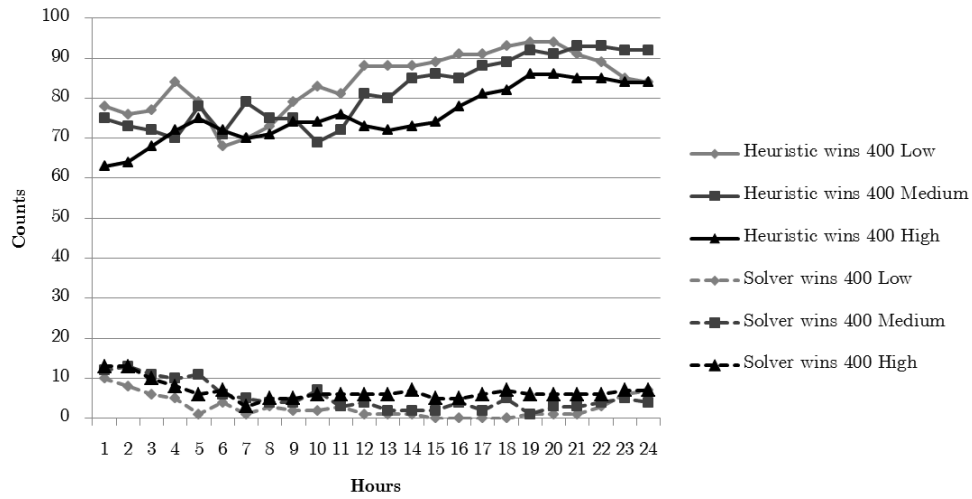


Figure 4-8 The trend of N+/N- for |I|=400 case across 24 hrs

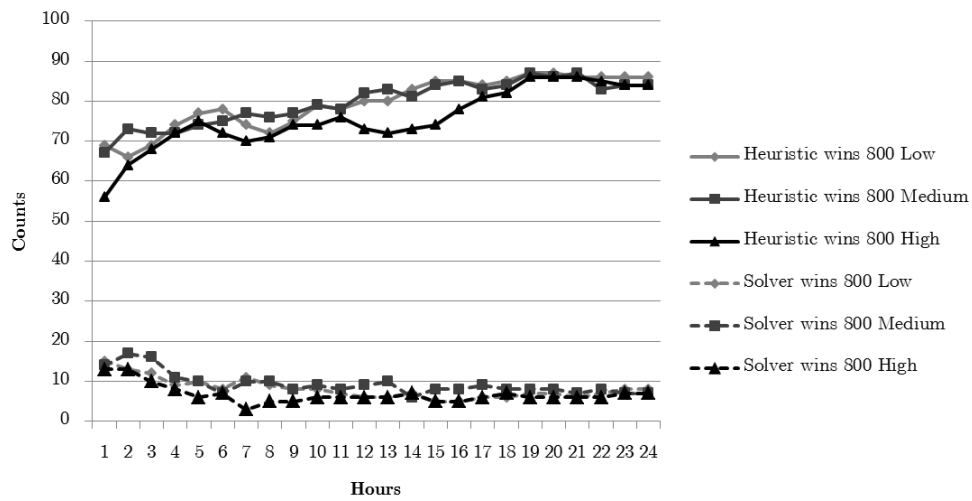


Figure 4-9 The trend of N+/N- for |I|=800 case across 24 hrs

Table 4-7 Results of the count of the success by our heuristic and solver by the grouping of time.

Hour	N^+ / N^-					
	$ I =400$			$ I =800$		
	Low	Medium	High	Low	Medium	High
1	78/10	75/12	63/13	69/15	67/14	56/13
2	76/8	73/13	64/13	66/13	73/17	64/13
3	77/6	72/11	68/10	69/12	72/16	68/10
4	84/5	70/10	72/8	74/9	72/11	72/8
5	79/1	78/11	75/6	77/10	74/10	75/6
6	68/4	71/6	72/7	78/8	75/7	72/7
7	70/1	79/5	70/3	74/11	77/10	70/3
8	73/3	75/4	71/5	72/9	76/10	71/5
9	79/2	75/4	74/5	75/8	77/8	74/5
10	83/2	69/7	74/6	79/8	79/9	74/6
11	81/3	72/3	76/6	78/7	78/8	76/6
12	88/1	81/4	73/6	80/6	82/9	73/6
13	88/1	80/2	72/6	80/6	83/10	72/6
14	88/1	85/2	73/7	83/7	81/6	73/7
15	89/0	86/2	74/5	85/5	84/8	74/5
16	91/0	85/4	78/5	85/5	85/8	78/5
17	91/0	88/2	81/6	84/6	83/9	81/6
18	93/0	89/5	82/7	85/6	84/8	82/7
19	94/1	92/1	86/6	87/7	87/8	86/6
20	94/1	91/3	86/6	87/7	86/8	86/6
21	91/1	93/3	86/6	86/6	87/7	86/6
22	89/3	93/4	85/6	86/7	83/8	85/6
23	85/6	92/5	84/7	86/8	84/7	84/7
24	84/7	92/4	84/7	86/8	84/7	84/7

Chapter 5 Fine-tuning on Timing Given the Module Reallocation Decision

5.1 Introduction

In chapter 3 and 4, we try to find a module reallocation solution such that it satisfies all the requirements from the stakeholders while the inter-campus traffic measurement is optimized. In terms of timing, we assume that it is given by the stakeholders. In reality, the solution obtained so far usually needs to be approved by the stakeholders. Several cases may happen: first, the objective value obtained so far is still unacceptable by the stakeholders so they agree to change the timing locally to even improve the inter-campus traffic measure. Second, as stakeholders may share different point of views and have different preferences, they may need more options to negotiate from each other, and solutions having an improved objective value should be a vital factor among their considerations. Such scenarios motivate the study in this chapter.

As this fine-tuning problem is different from our previous study, we first list some basic assumptions:

- We cannot change the set of modules being reallocated,
- We cannot change the timing of the modules which are not reallocated,
- We can change the timing and the room assignments of the modules.

The trivial assumption is that we cannot change the set of modules being reallocated anymore, as it provides us a vital starting point from which this chapter extends. On the other hand, the possible changeable decisions are the timing of modules and, further, the room assignments of the modules. The reason that we allow changes on room assignment is that changing timing may incur violations on type 2 constraints in MRPT. Revising the room assignment accordingly provides a possible way of reparation. Another assumption we set here is that we do not allow those changes which affect the old campus. The reason is described in chapter 1: In summary, there may exist too many intangible and unquantifiable constraints on the timetable from variant schools/departments. To identify and resolve the incurred issue, there could be too many human involvements needed for the stakeholders in the decision process. Intuitively, the fewer modules we consider, the less efforts in communicating and negotiating between the university management and individual school/department we need. Therefore, in this section we focus on only changing the timing of modules which are reallocated.

Another difference from this chapter to the previous one is that the decisions are in terms of class rather than module now, as a module may have different classes which are assigned to different timeslots. We therefore define the set C as all the classes and use decision variables x_{ct} to represent whether class c is reallocated and assigned to timeslot t , where $c \in C, t \in T$.

In the following section we describe our proposed fine-tuning method. Later, we show numerical experiment by illustrating several test cases as examples.

5.2 Methods of time-tuning

We propose a greedy heuristic method to conduct the fine-tuning on the timetable of modules which are reallocated according to the given module reallocation solution. This method iteratively conducts one-timeslot-exchange for candidate class in order to improve the objective value. As some one-timeslot-exchanges may violate the capacity constraint (3.11), additional timeslot-exchange or even room-type-exchange may be needed to ensure the resulting solution satisfies all aforementioned constraints. We describe the high level algorithm in Algorithm 5-1.

Algorithm 5-1: Fine-tuning on timing to improve inter-campus traffic

input The module reallocation solution X , the given timetable S

- 1 Search for the set of classes \hat{C} such that (1) they are in the worst moment and direction in the current solution and (2) they are reallocated.
- 2 **for all** classes c such that $c \in \hat{C}$ sorted descending by the corresponding traffic component value, **do**
- 3 Search for a new timing t' of c such that the new inter-campus traffic is strictly improved
- 4 **if** t' exists **then**
- 5 Apply t' and recalculate the worst moment and direction. **Go to** step 1.
- 6 **end if**
- 7 **end for**

output current timing and module reallocation solution

In this algorithm, the search space is the set of classes which contribute the highest inter-campus traffic of a specific direction with regard to a specific timeslot. Remind that the traffic contribution of timeslot t and direction d is measured by the overall traffic component $\sum_{i_1, i_2 \in I} r_{i_1 i_2 t}^n v_{i_1 i_2}$. As the objective function is in the mini-max form, we only need to investigate the highest overall traffic component and identify the target timeslot and direction. We identify the set of class-pairs which belongs to the chosen component. As explained before, we intent to change the timing for classes on the new campus only, so we narrow them down into set \hat{C} . We then rank these classes in a greedy way by sorting them in a descending order by the value of the corresponding traffic contribution. We investigate these classes one after another by conducting a neighborhood search. Among those of which the objective values are strictly better than a defined target z , i.e., $z > 0$, we choose the best as our new solution. We also re-identify the highest overall traffic contribution if the solution changes. The process is repeated until we cannot find any improving neighboring solution after exploring through \hat{C} .

In the following we describe the process of neighborhood search, summarized in Algorithm 5-2, in details. The neighborhood $\{\{X', S'\}\}$ to $\{X, S\}$ is defined as:

- S' is slightly different with S , (5.1),
- No timing conflict incurred by S' (5.2),

- X' is feasible with regard to MRPT (5.3).

To conduct the neighborhood search, we first try to reassign the timing according to (5.1) and (5.2). If we are able to find some, the best $\{X, S'\}$ becomes the solution. Otherwise there must be dissatisfaction in (5.3) as we do not change the room assignment yet. Under this circumstances, we can temporarily relax (5.3) by applying the reassignment of timing of c . The only possible violation incurred in MRPT is some capacity constraint, which means the number of classes assigned to that some timeslot exceeds the number of rooms available. Assuming the corresponding room type is \tilde{j} and the timeslot is \tilde{t} . To resolve the violation of capacity constraint, obviously there are two ways, i.e., change the room type or change the timing of every class which are assigned to \tilde{j} and \tilde{t} . The former method does not affect the improvement of objective value as long as the valid reassignment can be found, but the later may deteriorate the objective value. As a result, we conduct the room type reassignment first followed by timing reassignment for each target class. All generated new solutions are recorded as candidates and the best at the end is the output.

Algorithm 5-2: Searching for a neighboring solution from a specified class of a given timetable and module reallocation solution

input The class c , the timetable S , the module reallocation solution X and a target objective value z .

- 1 For c , find the set of new timeslots T_c which satisfy (5.2), (5.4) and $\Delta z > 0$.
 - 2 **if** $T_c \neq \emptyset$ **then**
 - 3 Apply the change of timing of c to t^* such that t^* leads to $\max \Delta F$.
Record the new solution.
 - 4 **else**
 - 5 Find the set of timeslots T_c' which satisfy (5.2) and $\Delta z > 0$ but violate (5.3)
 - 6 **if** $T_c' \neq \emptyset$ **then**
 - 7 **for all** $t \in T_c'$, denote the violated room type as \tilde{j} , **do**
 - 8 **if** we can find another valid room type for c **then**
 - 9 Call **Algorithm 5-1** with input (c, S, X with such room type change applied, z)
 - 10 **else**
 - 11 **for all** classes c' such that $x_{c,\tilde{j}} = 1$, **do**
 - 12 **if** we can find another valid room type for c' **then**
 - 13 Apply this room type change to X and record this new solution.
 - 14 **else**
 - 15 Call **Algorithm 5-1** with input (c', S with new timing t for c applied, X, z).
 - 16 **end if**
 - 17 **end for**
 - 18 **end if**
 - 19 **end for**
 - 20 **end if**
 - 21 **end if**
- output** the recorded solution with the best Δz .
-

For room type reassignment part, we try to find a new room type for the target class so that we obtain a feasible solution to MRPT. We call this reassignment as a valid reassignment. No matter whether the target class is lecture or tutorial, we check every possible one in the set of compatible room types which is defined in Chapter 3 to see whether it can be reassigned. If there is a valid room type reassignment, i.e., by applying this reassignment, the resulting room assignment satisfies all type 2 constraints, we obtain a neighboring solution.

If there does not exist a valid reassignment for the target class, we need to resort to the timing reassignment method. We use the idea of recursion to conduct the search: we check for every class which is in the corresponding room type and assigned to the problematic timeslot, and try to reassign it to another timeslot as long as the final benefit of improvement in objective value is positive. This timing reassignment is essentially the same to the neighbored search, with the only difference in terms of the timing decision part of the starting point.

In addition, we can also apply room type reassignment on c itself. In this case, the current violation can be avoided as we would not reassign to the current problematic room types. The search process can also reuse the current structure by changing the MRPT solution part of the starting point.

5.3 Numerical Experiment

We show one example on how the time-tuning procedure is conducted in this section. The purpose of this numerical experiment is to show the basic idea of the fine-tuning on timetable.

We used one result from 800-high case as the start point for fine-tuning on timetable. For this specific solution, we first obtained the inter-traffic evaluation for all five weekdays and two directions and illustrate them in Figure 5-1 (Direction A to B, where B is the new campus) and Figure 5-2 (Direction B to A). The worst moment and direction was 4pm on Friday from campus A to campus B and the corresponding traffic rate was 432.

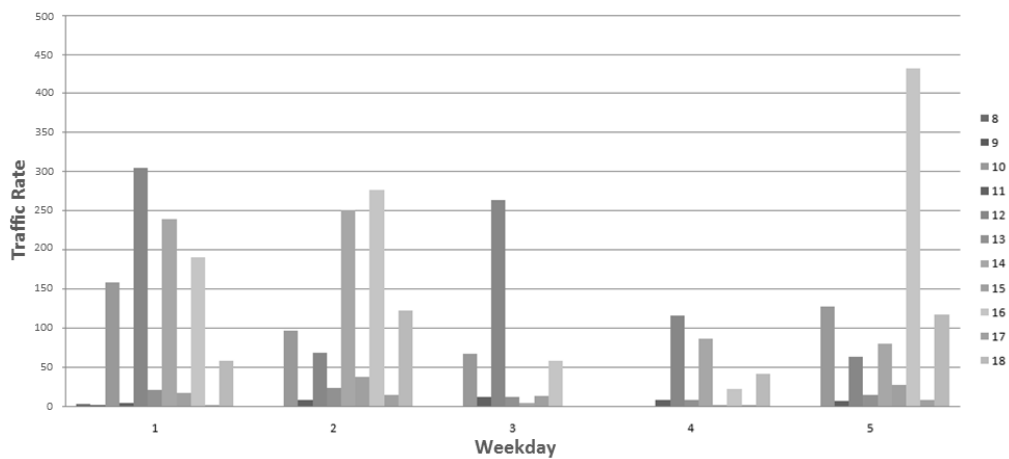


Figure 5-1 The detailed inter-campus traffic rate from campus A to campus B

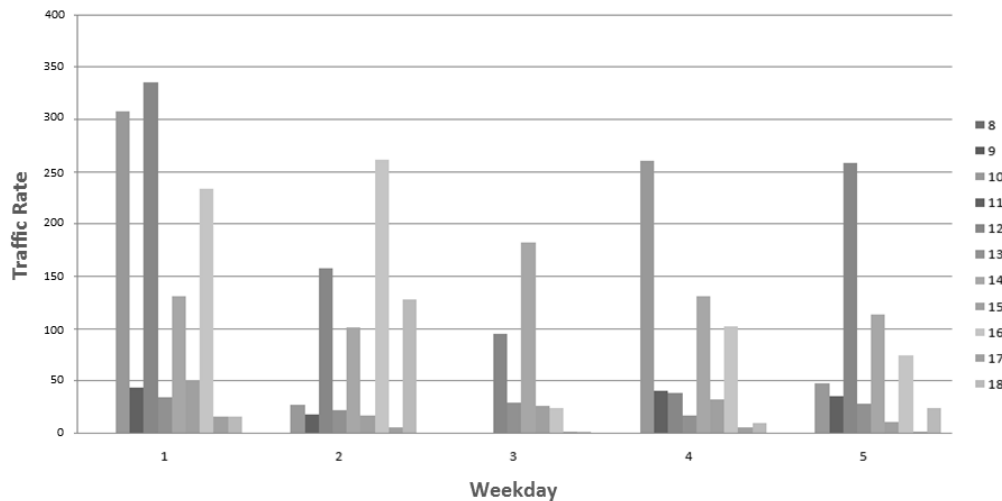


Figure 5-2 The detailed inter-campus traffic rate from campus B to campus A

Following the steps described in Algorithm 5-1, we sorted the class set \hat{C} , which are the candidates to conduct modification of timing, and checked each class one by one to see whether it is possible to change the corresponding timing. The first class in \hat{C} was class α . We found that if we rescheduled it to Thursday, no restrictions were violated.⁴² As for the impact on traffic rates on Friday, the traffic rate at 4pm from A to B was reduced by 101. In addition, other traffic rates on Friday from A to B were also affected with various reductions. As there was no student who took class α and also took classes afterwards on Friday, there were no such changes on traffic from B to A. On the other hand, traffic rate on Thursday was also affected by this rescheduling. Specifically, traffic rate at 4pm from A to B was increased by 52, totally

⁴² In fact there are several possible timeslots that we can reschedule and they all lead to the same reduction of objective value. We choose 4pm THU as the class hour remains the same as it is originally scheduled, which is preferred by stakeholders.

contributed by 52 students taking a course on campus A which ended at 3:45PM. No other changes on this direction were applied as there were no other classes which were correlated to α and ended before 4 PM. However, several classes, which were scheduled after 5PM, had a correlation with α . As a result, traffic rates from B to A after 5PM were increased accordingly. As an illustration, changes on traffic rate from A to B were shown in Figure 5-3, and the one from B to A is shown in Figure 5-4. In these figures, the blocks with solid outlines represent the traffic rates which are reduced, and the blocks with dashed outlines represent the traffic rates which are increased.

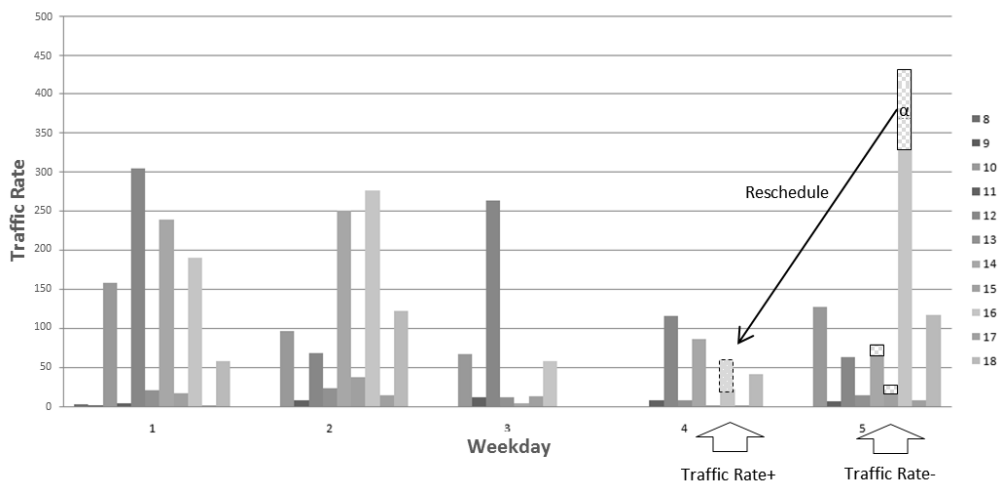


Figure 5-3 Changes on traffic rate (A to B) when rescheduling α from FRI to THU

After Algorithm 5-1 was terminated, we were able to reduce the objective value to 282, i.e., reduced the inter-campus traffic measure by about 35%. Although this significant reduction may not be achieved in every test case, it shows the potential of further improving inter-campus traffic.

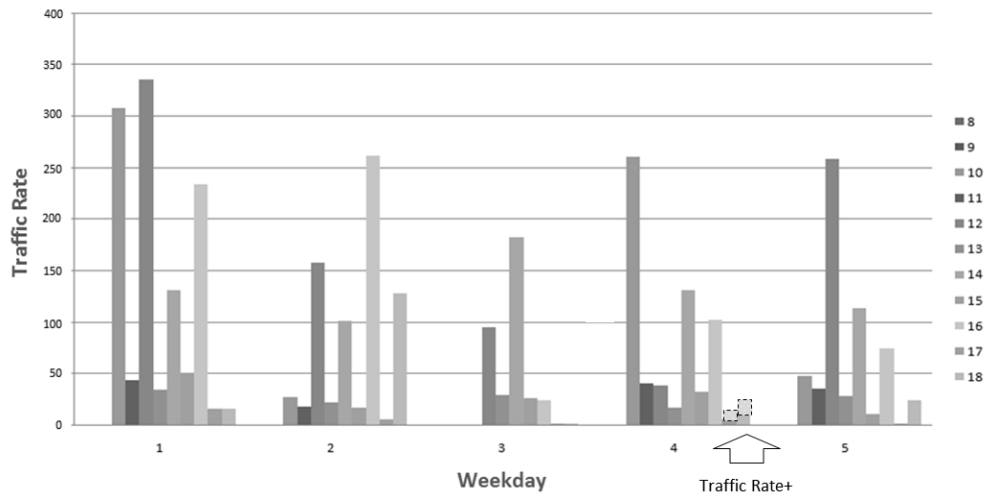


Figure 5-4 Changes on traffic rate (B to A) when rescheduling α from FRI to THU

Chapter 6 Conclusion

In this thesis, we have proposed the module reallocation problem, namely MRPT, which arises from the field of UCTP. As far as we know, this problem has never been studied before as it explicitly focuses on the impact on the inter-campus traffic. Given a timetable, the worst case scenario of inter-campus traffic is minimized subject to a set of stakeholders' requirements in order to decide which modules should be reallocated and which types of classrooms on the new campus should be assigned to. We have formulated this problem as a MIP model after conducting thorough data analysis on historical enrollment data and deriving all required model parameters. Solutions to the stakeholder's problem, which was a special and simplified MRPT, have been found by using the commercial solver CPLEX. However, we have also found that commercial solver is not able to obtain a good solution when the problem scale is large. As in reality the problem scale of MRPT is expected to be large, we have proposed an iterative two-stage heuristic to improve computational efficiency when finding good solutions. The two stages account for the decomposable decisions of the MRPT which were the module selection decision and the room allocation decision. This heuristic combines various methods, such as constructive heuristic, clustering analysis, branch and bound framework, Lagrangian relaxation method, etc., to exploit the problem structure and maintain the computational efficiency. We have solved various real-scale test cases of MRPT in which the number of modules are up to 800 and 10,000. Even when comparing with the previous studies on general UCTP,

the problem scale in our study is quite large. The results have shown that our proposed method outperforms CPLEX overwhelmingly. In addition, we have considered the case when we are allowed to modify the timing. As an extension to MRPT, we have provided a way to fine-tune the timetable to further improve the inter-campus traffic.

In Chapter 3, we have conducted data analysis to understand the problem better. First we have helped the stakeholders to find the connection between the students' movement behavior and the inter-campus traffic. We have also used cluster analysis to provide insights into how to prevent bad solutions. Second, we have helped the stakeholders to determine those parameters such as the target level of "fairness" for their requirement by exercising several what-if experiments. Through these very experiments, we have also found that these requirements prevent us from using trivial (good) solutions, such as assigning courses from the same faculty to the new campus. Third, we have formulated this real world problem as a MIP model by considering the stakeholders' requirements and analyzing related data. This MIP model was able to represent the stakeholders' needs including controlling the traffic while maintaining a set of constraints in terms of fairness. The solution to the problem that the stakeholders addressed was applied in reality after the aforementioned processes were completed. In terms of the consequences in reality, it is widely acknowledged by the stakeholders that our solution contributes a lot in terms of controlling the inter-campus traffic (jointly contributed by investing more shuttle buses as our solution cannot

reduce the traffic arbitrarily). On the other hand, by using commercial solver to solve this model, the commercial solver showed severe limitations when the problem scale is big, including not only low computational performance, but also difficulties in finding feasible solutions.

In Chapter 4, we have proposed a two-stage heuristic approach to solve this problem to handle the big problem scale. The two stages, namely the module selection stage and room assignment stage, are derived by exploiting the problem structures. In the first stage, we have introduced a multi-objective framework to tackle the problem. Under the multi-objective framework, we have proposed two methods to generate the solution. The first heuristic is a greedy constructive method based on the balancing between the objective value and the violations of constraints. The second heuristic constructs a bi-objective model, which uses a surrogate measure of traffic based on clustering analysis on the student-module registration data. This model is solved by the Normal Boundary Intersection (NBI) method. In the second stage, we have used a branch and bound framework to solve the problem. Within this framework, Lagrangian relaxation method is used to solve the sub-problem, in which a type of knapsack structure is identified and thus the sub-problem can be solved efficiently. We have also used constraint programming techniques to help obtain the incumbent solution. Comparing to the performance of CPLEX, the results show that the proposed method is able to provide solutions of good quality comparatively quicker.

In Chapter 5, we have further extended the MRPT by considering that the timetable is allowed to be modified slightly from any given one. We have conducted a local search from the existing solution to MRPT to a new solution with a modified timetable such that the inter-campus traffic measurement can be improved.

There are mainly two sets of results of this study. The first result set was obtained by solving a MRPT problem in a real life project, which was described in Chapter 3 and 5, by using commercial solver. The problem is a small-scale problem due to a lot of practical restraints encountered through the project. Solutions for two semesters of one academic year were provided to the stakeholders for the first year running of the new campus. The stakeholders appreciated them. In addition, important decisions such as investing more shuttle bus were made based on the results. It is also expected that the results can be more accurate as more recent data will be available in the future (comparing to the historical data which was used in our study). The second result set was generated by using the method we proposed in Chapter 4 on a variant set of large-scale generated input data based on the knowledge and experiences through the project. The results show that our proposed heuristic can improve the computational efficiency on large-scale problem which has not been addressed in our real life project.

Overall, we have studied a real-world problem closely related to UCTP. As far as we know, this problem has not been studied before. The problem is challenging in several aspects, including a very different objective function

namely inter-campus traffic, an innovative “facility-sharing” idea for the new campus and the large scale. We believe this study present a new direction of study related to UCTP and other university resource planning problems. We hope that our study provide guidance and insights for the related studies.

For the future studies, we propose several possible directions:

- (1) Our study considers the case when there are only two campuses of which one campus is in existence while the other is new. In reality, there are similar multi-campus module reallocation problem but may require very different ways to consider. For instance, some universities may have many campuses while more than two of them are required to be shared by students from all departments.⁴³ In this case, the MRPT is extended to three campuses and more. Another possibility is that no campus is new, i.e., all campuses are already in existence for several years.⁴⁴ In this case, the module reallocation problem may become a module exchange problem, as modules from each campus may be reallocated to other campus.

⁴³ For instance, the University of the Aegean (<http://www.aegean.gr/aegean2/index.html>), and the university of the Highlands and Islands (<http://www.uhi.ac.uk/en/#campuses>).

⁴⁴ For instance, University of Manchester with University of Manchester Institute of Science and Technology (<http://www.manchester.ac.uk/discover/history-heritage/history/>).

(2) The concept of inter-campus traffic under campus extension can be applied in other circumstances. For instance, it may help to make decisions for the case of hospital extension. In china, it becomes critical that the medical resources, especially those major hospitals, cannot satisfy the dramatically increasing demand of the patients. As a result, some of the major hospitals are now upgrading by building a new complex next to the original one. As for the location of the new complex, it is usually not possible to build very nearby since the major hospitals are typically built years ago and the surroundings are already taken by others. However, it is also not possible to be located remotely, as many diseases are correlated and the common facilities (e.g., laboratories, imaging centers, the emergency) are most likely to be remained in the original complex due to limited resources. The common way in china for the hospital extension project is to buy some land over the street and build the new one which is linked by an underground passageway. The doctors, however, usually serve both complexes under a schedule. For example, on Monday a doctor of hematology may need to be at his office from 9AM to 10AM, and visits the ward for patients of the clinic of cardiology after that till 1PM, then works back at his office till 5PM, and stays at his station of surgery in the night. However, the travelling times for him at different times of the day are usually dramatically different. The main reason is that the average number of patient-visitings per day is huge in china. It is common to even have congestion in the underground passageway,

and the queue for a lift is usually extremely long. The other minor reasons include misleading guidance system, rush hours (e.g., children are tend to visit the hospital in the noon as that is the only time available for the escorting parents) and so forth. As a result, the different location arrangements of a doctor's office and the other stations he is responsible can affect their travelling time a lot, and hence affect their work performance (e.g., late for their next station). Different offices and stations are also correlated in such a way that they are used by doctors travelling from different locations. In this case, a smooth traffic for doctors between the two complexes is obviously favored. This problem is very similar to our campus extension case. The obvious difference is that the facilities of a same clinic/department should be at the same complex. However, the wards, for example, for different patients are allowed to be placed into different complexes. One may even consider the case for the patients and can also consider whether the schedule for the doctors should be fixed in the first place.

- (3) One of the extensions to this study can be the shuttle bus dispatching system. In this case, the relatively stable demand from students is presented by a fixed module selection and timetabling. An intelligent dispatching system can be connected directly and produce a more efficient dispatch plan.
- (4) As we mentioned in Chapter 3, our model relies on historical data when it was developed. Therefore the results from our model may be

different from the real traffic. We believe this can be resolved by running this model for several years, as the new input data, which reflects the students' responding to the actual inter-campus traffic system, are used. Certain parameters, such as the overlap value, can be fine-tuned by considering both the enrollment data and the real traffic data. It is also good if some survey can be conducted in the end of a semester to collect student's opinion on the traffic impact from the module reallocation decision. Moreover, when more data are available, we could even build stochastic model to capture students' behavior under multiple scenarios. For instance, certain modules are constantly popular from year to year, and students may tend to avoid selecting such modules if possible as he expect a high traffic when attending this module.

- (5) Our fine-tuning study in Chapter 5 serves as an extension to the main study which proposes a simple greedy approach. One may consider a more complete approach by considering more constraints such as SC3 mentioned in Chapter 2. In that case, one can add corresponding search rules to Algorithm 5-2.

Bibliography

- Agustín-Blas, L.E., S. Salcedo-Sanz, E.G. Ortiz-García, A. Portilla-Figueras, and Á.M. Pérez-Bellido. 2009. "A hybrid grouping genetic algorithm for assigning students to preferred laboratory groups." *Expert Systems with Applications* no. 36 (3):7234-7241.
- Avella, Pasquale, and Igor Vasil'Ev. 2005. "A Computational Study of a Cutting Plane Algorithm for University Course Timetabling." *Journal of Scheduling* no. 8 (6):497-514. doi: 10.1007/s10951-005-4780-1.
- Ayob, M., and G. Kendall. 2003. A monte carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine.
- Back, Thomas, David B Fogel, and Zbigniew Michalewicz. 1997. *Handbook of evolutionary computation*: IOP Publishing Ltd.
- Bilgin, B., E. Özcan, and E. Korkmaz. 2007. "An experimental study on hyper-heuristics and exam timetabling." *Practice and Theory of Automated Timetabling VI*:394-412.
- Burke, E., M. Dror, S. Petrovic, and R. Qu. 2005. "Hybrid graph heuristics within a hyper-heuristic approach to exam timetabling problems." *The next wave in computing, optimization, and decision technologies*:79-91.
- Burke, E. K., G. Kendall, and E. Soubeiga. 2003. "A Tabu-Search Hyperheuristic for Timetabling and Rostering." *Journal of Heuristics* no. 9 (6):451-470. doi: 10.1023/B:HEUR.0000012446.94732.b6.
- Burke, E. K., J. Marecek, A. J. Parkes, and H. Rudová. 2010. "Decomposition, reformulation, and diving in university course timetabling." *Computers &*

- Operations Research* no. 37 (3):582-597. doi: DOI: 10.1016/j.cor.2009.02.023.
- Burke, E., R. Qu, and A. Soghier. 2012. "An Adaptive Tie Breaking and Hybridisation Hyper-Heuristic for Exam Timetabling Problems." *Nature Inspired Cooperative Strategies for Optimization (NICSO 2011)*:205-223.
- Burke, E.K., S. Petrovic, and R. Qu. 2006. "Case-based heuristic selection for timetabling problems." *Journal of Scheduling* no. 9 (2):115-132.
- Burke, Edmund K, Adam J Eckersley, Barry McCollum, Sanja Petrovic, and Rong Qu. 2010. "Hybrid variable neighbourhood approaches to university exam timetabling." *European Journal of Operational Research* no. 206 (1):46-53.
- Burke, Edmund K., Barry McCollum, Amnon Meisels, Sanja Petrovic, and Rong Qu. 2007. "A graph-based hyper-heuristic for educational timetabling problems." *European Journal of Operational Research* no. 176 (1):177-192. doi: DOI: 10.1016/j.ejor.2005.08.012.
- Burke, Edmund, Jakub Mareček, Andrew Parkes, and Hana Rudová. 2012. "A branch-and-cut procedure for the Udine Course Timetabling problem." *Annals of Operations Research* no. 194 (1):71-87. doi: 10.1007/s10479-010-0828-5.
- Byskov, J.M. 2004. "Enumerating maximal independent sets with applications to graph colouring." *Operations Research Letters* no. 32 (6):547-556.
- Campêlo, M., R. Corrêa, and Y. Frota. 2004. "Cliques, holes and the vertex coloring polytope." *Information Processing Letters* no. 89 (4):159-164.

- Caprara, Alberto, Hans Kellerer, Ulrich Pferschy, and David Pisinger. 2000. "Approximation algorithms for knapsack problems with cardinality constraints." *European Journal of Operational Research* no. 123 (2):333-345.
- Carrasco, Marco, and Margarida Pato. 2001. "A Multiobjective Genetic Algorithm for the Class/Teacher Timetabling Problem Practice and Theory of Automated Timetabling III." In, edited by Edmund Burke and Wilhelm Erben, 3-17. Springer Berlin / Heidelberg.
- Carter, Michael W, Gilbert Laporte, and John W Chinneck. 1994. "A general examination scheduling system." *Interfaces* no. 24 (3):109-120.
- Ceschia, S., L. Di Gaspero, and A. Schaerf. 2011. "Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem." *Computers & Operations Research*.
- Chu, Paul C, and John E Beasley. 1998. "A genetic algorithm for the multidimensional knapsack problem." *Journal of heuristics* no. 4 (1):63-86.
- Corne, D., P. Ross, and H.L. Fang. 1994. Evolutionary timetabling: Practice, prospects and work in progress.
- Cowling, P., G. Kendall, and E. Soubeiga. 2001. A parameter-free hyperheuristic for scheduling a sales summit.
- Das, Indraneel, and John Dennis. 1996. Normal-Boundary Intersection: An Alternate Method for Generating Pareto Optimal Points in Multicriteria Optimization Problems. DTIC Document.
- Daskalaki, S., T. Birbas, and E. Housos. 2004. "An integer programming formulation for a case study in university timetabling." *European Journal of*

- Operational Research* no. 153 (1):117-135. doi: 10.1016/s0377-2217(03)00103-6.
- Davis, Lawrence. 1991. "Order-Based Genetic Algorithms and the Graph Coloring Problem." In *Handbook of genetic algorithms*, 72-90. New York: Van Nostrand Reinhold.
- De Causmaecker, Patrick, Peter Demeester, and Greet Vanden Berghe. 2009. "A decomposed metaheuristic approach for a real-world university timetabling problem." *European Journal of Operational Research* no. 195 (1):307-318.
- Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan. 2002. "A fast and elitist multiobjective genetic algorithm: NSGA-II." *Evolutionary Computation, IEEE Transactions on* no. 6 (2):182-197. doi: 10.1109/4235.996017.
- Delmaire, Hugues, JUAN A Diaz, ELENA Fernandez, and MARUJA Ortega. 1999. "Reactive GRASP and tabu search based heuristics for the single source capacitated plant location problem." *Infor-Information Systems and Operational Research* no. 37 (3):194-225.
- Deris, Safaai, Sigeru Omatu, Hiroshi Ohta, and Puteh Saad. 1999. "Incorporating constraint propagation in genetic algorithm for university timetable planning." *Engineering Applications of Artificial Intelligence* no. 12 (3):241-253. doi: Doi: 10.1016/s0952-1976(99)00007-x.
- Di Gaspero, L., B. McCollum, and A. Schaerf. 2007a. The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3).

- Di Gaspero, Luca, Barry McCollum, and Andrea Schaerf. 2007b. The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3). Paper read at Proceedings of the 14th RCRA workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion, Rome, Italy.
- Di Gaspero, Luca, and Andrea Schaerf. 2001. "Tabu Search Techniques for Examination Timetabling." In *Practice and Theory of Automated Timetabling III*, edited by Edmund Burke and Wilhelm Erben, 104-117. Springer Berlin / Heidelberg.
- Dowland, K.A., E. Soubeiga, and E. Burke. 2007. "A simulated annealing based hyperheuristic for determining shipper sizes for storage and transportation." *European Journal of Operational Research* no. 179 (3):759-774.
- Ehrgott, Matthias. 2006. "A discussion of scalarization techniques for multiple objective integer programming." *Annals of Operations Research* no. 147 (1):343-360.
- Eiben, A. E., J. K. van der Hauw, and J. I. van Hemert. 1998. "Graph Coloring with Adaptive Evolutionary Algorithms." *Journal of Heuristics* no. 4 (1):25-46. doi: 10.1023/a:1009638304510.
- Erben, W. 2001. "A grouping genetic algorithm for graph colouring and exam timetabling." *Practice and Theory of Automated Timetabling III*:132-156.
- Ersoy, E., E. Özcan, and Ş. Uyar. 2007. Memetic algorithms and hyperhill-climbers.

- Falkenauer, E. 1999. "Applying genetic algorithms to real-world problems." *IMA Volumes In Mathematics And Its Applications* no. 111:65-88.
- Falkenauer, Emanuel. 1997. *Genetic algorithms and grouping problems*. New York :: Wiley.
- Fisher, Marshall L. 1981. "The Lagrangian Relaxation Method for Solving Integer Programming Problems." *MANAGEMENT SCIENCE* no. 27 (1):1-18.
- Gaivoronski, A. 1988. *Stochastic quasigradient methods and their implementation*. Springer-Verlag Berlin, Germany.
- Galati, M. 2010. *Decomposition methods for integer linear programming*. 3389951, Lehigh University, United States -- Pennsylvania.
- Geiger, Martin. 2009. "Multi-criteria Curriculum-Based Course Timetabling—A Comparison of a Weighted Sum and a Reference Point Based Approach Evolutionary Multi-Criterion Optimization." In, edited by Matthias Ehrgott, Carlos Fonseca, Xavier Gandibleux, Jin-Kao Hao and Marc Sevaux, 290-304. Springer Berlin / Heidelberg.
- Gotlieb, CC. 1962. The construction of class-teacher time-tables. Paper read at COMMUNICATIONS OF THE ACM.
- Gottlieb, Jens. 2000. On the effectivity of evolutionary algorithms for the multidimensional knapsack problem. Paper read at Artificial Evolution.
- Han, L., and G. Kendall. 2003. An investigation of a Tabu assisted hyper-heuristic genetic algorithm.

- Hastie, Trevor, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. 2013. *The elements of statistical learning*. 2 ed. Vol. 2: Springer.
- Holmberg, Kaj, and Di Yuan. 2000. "A Lagrangian heuristic based branch-and-bound approach for the capacitated network design problem." *Operations Research* no. 48 (3):461-481.
- Hwang, Ching-Lai, and Abu Syed Md Masud. 1979. *Multiple objective decision making—methods and applications*.
- Ismayilova, Nergiz A., Mujgan Sağır, and Rafail N. Gasimov. 2007. "A multiobjective faculty–course–time slot assignment problem with preferences." *Mathematical and Computer Modelling* no. 46 (7–8):1017-1029. doi: 10.1016/j.mcm.2007.03.012.
- Jiang, Dongchen, and Tobias Nipkow. 2013. "Hall's marriage theorem." *The Archive of Formal Proofs (December 2010)*, <http://afp.sf.net/entries/Marriage.shtml>.
- Kannan, R., S. Vempala, and A. Veta. 2000. On clusterings-good, bad and spectral. Paper read at Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on, 2000.
- Kellerer, Hans, Ulrich Pferschy, and David Pisinger. 2004. *Knapsack problems*: Springer.
- Kendall, G., and M. Mohamad. 2004. Channel assignment in cellular communication using a great deluge hyper-heuristic.
- Kendall, G., E. Soubeiga, and P. Cowling. 2002. Choice function and random hyperheuristics.

- Konak, A., D.W. Coit, and A.E. Smith. 2006. "Multi-objective optimization using genetic algorithms: A tutorial." *Reliability Engineering & System Safety* no. 91 (9):992-1007.
- Lach, Gerald, and Marco Lübbecke. 2008. "Optimal University Course Timetables and the Partial Transversal Polytope Experimental Algorithms." In, edited by Catherine McGeoch, 235-248. Springer Berlin / Heidelberg.
- Lach, Gerald, and Marco Lübbecke. 2012. "Curriculum based course timetabling: new solutions to Udine benchmark instances." *Annals of Operations Research* no. 194 (1):255-272. doi: 10.1007/s10479-010-0700-7.
- Lewis, R. 2008. "A survey of metaheuristic-based techniques for university timetabling problems." *OR Spectrum* no. 30 (1):167-190.
- Lewis, R., and B. Paechter. 2007. "Finding Feasible Timetables Using Group-Based Operators." *Evolutionary Computation, IEEE Transactions on* no. 11 (3):397-413.
- Lewis, R., B. Paechter, and B. McCollum. 2007. "Post enrolment based course timetabling: A description of the problem model used for track two of the second international timetabling competition." *Accounting and Finance Section*.
- Lewis, Rhydian, and Ben Paechter. 2005. "Application of the Grouping Genetic Algorithm to University Course Timetabling Evolutionary Computation in Combinatorial Optimization." In, edited by Günther Raidl and Jens Gottlieb, 144-153. Springer Berlin / Heidelberg.

- Martí, Rafael, Vicente Campos, MAURICIO GC Resende, and ABRAHAM Duarte. 2011. "Multi-objective grasp with path-relinking." *AT&T Labs Research Technical Report*.
- Messac, Achille, Amir Ismail-Yahaya, and Christopher A Mattson. 2003. "The normalized normal constraint method for generating the Pareto frontier." *Structural and multidisciplinary optimization* no. 25 (2):86-98.
- Michael, R Garey, and S Johnson David. 1979. "Computers and intractability: a guide to the theory of NP-completeness." *WH Freeman & Co., San Francisco*.
- MirHassani, S., and F. Habibi. 2011. "Solution approaches to the course timetabling problem." *Artificial Intelligence Review*:1-17. doi: 10.1007/s10462-011-9262-6.
- Motta, Renato de S, Silvana MB Afonso, and Paulo RM Lyra. 2012. "A modified NBI and NC method for the solution of N-multiobjective optimization problems." *Structural and Multidisciplinary Optimization* no. 46 (2):239-259.
- Murray, K., and T. Müller. 2007. Real-time student sectioning.
- Nareyek, A. 2003. "Choosing search heuristics by non-stationary reinforcement learning." *Applied Optimization* no. 86:523-544.
- Norvig, Peter. 1992. *Paradigms of artificial intelligence programming: case studies in Common LISP*: Morgan Kaufmann.
- Papoutsis, K., C. Valouxis, and E. Housos. 2003. "A Column Generation Approach for the Timetabling Problem of Greek High Schools." *The Journal of the Operational Research Society* no. 54 (3):230-238.

- Perzina, Radomír. 2007. "Solving the University Timetabling Problem with Optimized Enrollment of Students by a Self-adaptive Genetic Algorithm Practice and Theory of Automated Timetabling VI." In, edited by Edmund Burke and Hana Rudová, 248-263. Springer Berlin / Heidelberg.
- Pillay, N., and W. Banzhaf. 2009. "A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem." *European Journal of Operational Research* no. 197 (2):482-491. doi: DOI: 10.1016/j.ejor.2008.07.023.
- Polyak, Boris T. 1967. "A general method of solving extremum problems." *Doklady Akademii Nauk SSSR* no. 174 (1):33-&.
- Puchinger, Jakob, Günther R Raidl, and Ulrich Pferschy. 2010. "The multidimensional knapsack problem: Structure and algorithms." *INFORMS Journal on Computing* no. 22 (2):250-265.
- Qu, R., and E. Burke. 2005. "Hybrid variable neighborhood hyperheuristics for exam timetabling problems."
- Qu, R., E. Burke, B. McCollum, L. Merlot, and S. Lee. 2009. "A survey of search methodologies and automated system development for examination timetabling." *Journal of Scheduling* no. 12 (1):55-89. doi: 10.1007/s10951-008-0077-5.
- Qu, Rong, Edmund K. Burke, and Barry McCollum. 2009. "Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems." *European Journal of Operational Research* no. 198 (2):392-404. doi: DOI: 10.1016/j.ejor.2008.10.001.

- Qualizza, A., and P. Serafini. 2005. "A column generation scheme for faculty timetabling." *Practice and Theory of Automated Timetabling V*:161-173.
- Raidl, Günther R, and Jens Gottlieb. 2005. "Empirical analysis of locality, heritability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem." *Evolutionary Computation* no. 13 (4):441-475.
- Ross, Peter, Emma Hart, and Dave Corne. 1998. "Some observations about GA-based exam timetabling
Practice and Theory of Automated Timetabling II." In, edited by Edmund Burke and Michael Carter, 115-129. Springer Berlin / Heidelberg.
- Schaerf, A. 1999. "A survey of automated timetabling." *Artificial Intelligence Review* no. 13 (2):87-127. doi: 10.1023/a:1006576209967.
- Schaerf, Andrea, and Luca Di Gaspero. 2007. "Measurability and Reproducibility in University Timetabling Research: Discussion and Proposals
Practice and Theory of Automated Timetabling VI." In, edited by Edmund Burke and Hana Rudová, 40-49. Springer Berlin / Heidelberg.
- Sen, S, and Hanif D Sherali. 1986. "A class of convergent primal-dual subgradient algorithms for decomposable convex programs." *Mathematical Programming* no. 35 (3):279-297.
- Shukla, Pradyumn Kumar. 2007. "On the normal boundary intersection method for generation of efficient front." In *Computational Science-ICCS 2007*, 310-317. Springer.

- Stallaert, Jan. 1997. "Automated Timetabling Improves Course Scheduling at UCLA." *Interfaces* no. 27 (4):67-81.
- Thompson, Jonathan, and Kathryn Dowsland. 1996. "Variants of simulated annealing for the examination timetabling problem." *Annals of Operations Research* no. 63 (1):105-128. doi: 10.1007/bf02601641.
- Tripathy, Arabinda. 1984. "School Timetabling--A Case in Large Binary Integer Linear Programming." *MANAGEMENT SCIENCE* no. 30 (12):1473-1489. doi: 10.1287/mnsc.30.12.1473.
- Ulungu, EL, and J Teghem. 1994. "Multi - objective combinatorial optimization problems: A survey." *Journal of Multi - Criteria Decision Analysis* no. 3 (2):83-104.
- White, George M., and Chan Pak-Wah. 1979. "TOWARDS THE CONSTRUCTION OF OPTIMAL EXAMINATION SCHEDULES." *INFOR* no. 17 (3):219-229.
- White, George M., and Junhan Zhang. 1998. "Generating Complete University Timetables by Combining Tabu Search with Constraint Logic." In *Practice and Theory of Automated Timetabling II*, edited by Edmund Burke and Michael Carter, 187. Springer Berlin / Heidelberg.
- Wolsey, Laurence A. 1998. *Integer programming*. New York :: Wiley.
- Yang, S., and S.N. Jat. 2011. "Genetic algorithms with guided and local search strategies for university course timetabling." *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* no. 41 (1):93-106.
- Zhou, Rong, and Eric A Hansen. 2005. Beam-Stack Search: Integrating Backtracking with Beam Search. Paper read at ICAPS.

Zitzler, Eckart, Marco Laumanns, Lothar Thiele, Eckart Zitzler, Eckart Zitzler, Lothar Thiele, and Lothar Thiele. 2001. SPEA2: Improving the strength Pareto evolutionary algorithm. Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK).

Appendices

A.1 Test Case Generation for Numerical Experiment

A test case reflects a randomly generated timetabling environment of a university. For instance, how many modules are offered, how many students are involved, how many rooms are allocable, etc. A test case can be used to evaluate the quality of a module reallocation solution, such as traffic level. The whole test case generation process includes four steps. First, we generate the modules and associated classes (lectures and tutorials). Second, we generate the classrooms on the new campus. Third, we generate a timetable for all classes which will be used in the process of the module reallocation. Forth, we generate students according to the scale of the modules and assign them to classes. This process is designed based on the actual module selection operations of an Asian leading university which we study at.

Step I: Module generation

In the first step, we mainly generate the set of modules and other relevant information., such as the features related to modules, including module size, group of students eligible to select this module, etc., and the configuration of modules including associated lectures and tutorials, the number of lectures/tutorials per week of a module, etc. The set of modules (I) we consider are the candidates to relocate to the new campus and are initially generated. Then the set faculties (U) are generated based on $|I|$. For instance, 500 modules and three faculties are generated for a medium-sized university

test case. Given I , we then generate the features of each module in order to build a realistic test case. These features include: (1) the capacity of the module; (2) the faculty that offers the module; (3) the group of students eligible to select the module. Generation of these features is guided by several pre-defined patterns that are commonly seen in universities. We list the rules that we use as an example in Table 6-1.

Table 6-1 Rules used to generate features of modules

Category	Rules of feature generation	Example
Capacity of module	As the range of capacity of modules is usually large, it is split into α groups. First, we use a discrete random variable Y from 1 to α to randomly determine which group the capacity of the module would fall in. The probability mass function of the random variable is decreasing from 1 to α . Then the actual capacity of a module is randomly selected within the selected group.	Module capacity is in the range of [20,500], which is further categorized into four (i.e., $\alpha = 4$) groups: 20-50, 51-100, 101-250, 251-500. $\rho_Y(y)$ are 40%, 30%, 20%, 10% ⁴⁵ respectively when $y = 1, 2, 3, 4$.
The faculty offering the module	Randomly assigned to one faculty following a pre-defined distribution which represents the involvements among different faculties.	$\frac{1}{3}$ for each faculty, assuming $ U = 3$.
Eligible faculties where the students come from)	Compulsory module: Only the faculty that opens this module.	40% chance
	Selective module: All faculties.	60% chance
Eligible students' level ⁴⁶	Compulsory modules: Randomly be assigned to one of the groups of students following a pre-defined distribution which represents the involvements among student's level.	Four groups: level 1, level 2, level 3 and level 4. The probability of each group is equal.
	Selective modules: Students from all faculties.	Eligible for all students.

⁴⁵ Most university provides more selective modules than compulsory modules, and most of the selective modules are naturally small ones.

⁴⁶ Assuming all undergraduate students have four years of study. In this case, level 1 refers to a first-year student and "level 4" refers to a fourth year student.

After the features of modules being generated, we construct the configuration of the modules. For every module, we randomly generate its lectures and tutorials based on the module size. In reality, lectures are usually relatively few (1-2 lectures of a module per week) but large sized and tutorials are most likely many but small sized. The number of lectures and tutorials of a module per week vary mainly due to different module sizes. Although there is no tutorial for small modules, the number of tutorials of other modules is generally only determined by the module size, as the size of a tutorial is basically fixed to a small one, for instance, 20-30 students. However, the number of lectures of a module is also determined by other factors, such as teaching plans and teachers' preferences, and such factors are too complicated to capture. Instead, we introduce some simplified rules learned from historical data to generate realistic data. Each class taker may either take two lectures in a week, or take one of the two classes which split the whole takers into two groups. Length of each tutorial is one hour. Length of lecture may be either one hour or two hours. We show such patterns in table 3.

Table 6-2 Configuration of modules

Category	Scenario	Number of class(es) per week	Length of class periods
Lecture configuration	For small-sized module	1 lecture	2 hours ⁴⁷
	For medium-sized module	2 lectures. Each with all takers	1-1 ⁴⁸ (50% chance) or 1-2 (50%).
	For large-sized module	(1) (50% chance) 2 lectures. Each with half takers	1-1 or 2-2
		(2) (50% chance) 2 lectures. Each with all takers	1-1 or 1-2
Tutorial configuration	For small sized modules	No tutorial	n/a
	For medium and large-sized modules	As long as that one tutorial session contains 20 to 30 students.	1 hour long.

Step II: Room generation

We then generate information related to classrooms of the new campus. We first generate J based on the classes generated previously, as in reality the design of the classrooms of a new campus should consider information of previous timetabling system. The number of room types depends on the range of class sizes in the test case, as well as the percentage of capacity to facilitate the students of the whole university (in our case we use 30%). The wider the range of class size is the more room types are generated. The room size of one

⁴⁷ There is a ten to fifteen minutes break in the middle of the class.

⁴⁸ Format in A-B, where A means the length of the first class and B means the length of the second class (in hours).

type is an increasing nonlinear function of room type index. In addition, the number of rooms is generated similarly as a decreasing nonlinear function. (For instance, assuming $|J|=10$, $g_j = \lceil 5j^2 + 10 \rceil$, $k_j = \lceil 0.2j^2 - 4j + 22 \rceil$ where g_j and k_j is the room size and the number of rooms of type j respectively) These settings reflect the fact from reality that small rooms are generally plenty for the usage of all tutorials and many small modules, whereas large rooms are rare but necessary for the usage of several large lectures.

Step III: Timing generation

Recall that we do not change the original timing of modules when we do the module selection, so we need to generate the timing for all modules first. We do not generate the rooms for the original campus. Instead, we use a set of rules to assign classes into timeslots assuming the rooms are always sufficient. These rules intend to balance the utilization of timeslots as well as rooms. Classes are first divided into three groups: Large classes, medium classes and small classes. Due to our settings, all tutorials are treated as small classes. We assign classes to timeslots in the following order: large lecture pairs, large single lecture, medium lecture pairs, medium single lecture, small lecture, tutorials. Orders of classes in each sub group are random. In addition, each timeslot has three counters recording the number of large/medium/small classes. When a class is assigned to a timeslot, we update the corresponding counter. Each timeslot also has $|U|$ counters to record the number of classes

assigned from each faculty. We use 60 timeslots covering five working days. To generate the timetable, every class is randomly assigned into the timeslots (or some subset of timeslots, which is explained in the following section) which have the lowest corresponding counter value (As for a class pair, such as two lectures of a module, two timeslots satisfying the pre-defined preference are assigned.) and the lowest corresponding faculty counter value. As a result, classes in each timeslot should have similar class scales, and classes from the same faculty should be spread out in the whole week. This is often crucial for a good timetable as it brings fewer difficulties for students to register their preferred modules.

As many modules have a lot of tutorials associated, it is highly not possible that a student cannot select two modules because of timing confliction of lectures but of tutorials. Therefore, it is also preferred to assign lectures and tutorials to two different sets of timeslots. One possible way is that assign lectures to odd timeslots (assuming the first timeslot of a day is timeslot 1) but tutorials to even timeslots. To further refine this method, those 2-hour lectures are assigned to some specific time period such that no tutorials are allowed to be assigned to.

Timings for the lectures and tutorials are mainly assigned randomly. However, those modules having two lectures per week will be mostly assigned to two distinct days by randomly choosing one of the some patterns, e.g., Monday and Wednesday, Tuesday and Thursday, etc. We then use 60 timeslots covering five working days to construct T . Timing for lectures may

follow repetitive patterns like Monday and Wednesday or Tuesday and Thursday if there are two lectures per week. Timing for tutorials are chosen right after or one or two hours after the lectures. Each module has the equal opportunity to be tagged as one of the three faculties.

Step IV: Students generation

With timing of each class confirmed, we generate students and assign them to those classes. As every student must register a certain amount of compulsory modules according to his student's level and the faculty he belongs to, we consider compulsory module assignment first. Then we can have more freedom to assign them into selective modules. In the compulsory module assignment phase, all students are generated. Once a student is generated, he is immediately assigned to several compulsory modules. In the selective module assignment phase, every student is randomly assigned to several selective modules. This sequence reflects the priority usually used when students are selecting modules, as compulsory modules are much more important, and the pattern of selective module selection is often random.

We describe the process of compulsory module selection in more details. Note that any compulsory modules are assumed to be eligible to only one faculty and one student's level. The generation process is described in the following:

Step 1: A new student will be generated if there exists a module (called active module) in which the number of assigned takers so far has not reached

the module size and not tagged. Denote the newly generated student as student n . If no new student is needed the process stops.

Step 2: n is randomly assigned to one of the student levels that have at least one active module. n is also randomly assigned to one of the faculties opening those active modules of his level. Therefore, his student level and origin of faculty is determined.

Step 3: n is randomly assigned to one of his eligible modules repeatedly which bring no timing conflict, and the probability of choosing each module is proportional to the module size, as long as he selects enough modules according to his required amount (Denoted as α). If no more modules can be assigned and the number of selected modules has not met the requirement, go to step 4; otherwise, go to step 1.

Step 4: Choose the largest module (denoted as module A) among those assigned to n so far (Denoted as set of module Π). Try to find a module (denoted as module B) not in Π in which there exists a student (denoted as student x) who does not select module A . Let student x choose module A and therefore allow n select module B . Repeat this process until n can select enough modules. If success, go to step 1; If fails, go to step 5.

Step 5: Tag those modules which have not enough takers so far. Decrease α according to a predefined amount (e.g. 60%). Then keep generating students of the same student's level of n , and try to assign them to the remaining modules using the exact order of modules with more empty

seats first. If there are still seats left for some modules, update the associated module size. Go to step 1.

This aforementioned process tries to generate students so that every compulsory module has no empty seat and the number of students generated is not too big. The probability settings used in step 3 and the process in step 4 prevent the case when the larger modules have many seats not assigned. As the number of modules which has not enough takers is bounded by the sum of the requirement number of modules across different faculties and student levels, the number of students may not select enough modules should be controlled. On the other hand, the case students cannot select enough modules happens frequently in reality.

The process of selective module assignment is similar in some extend. Note that there is no origin of faculty restriction now, but there is still restriction on student's level. We reuse the process when we assign compulsory modules, except that (1) in step 2 there is no minimum required number of selections; (2) step 4 and 5 are no longer needed. We still use the probability settings in step 2 because usually large selective modules are more popular than smaller ones.

With the information generated, we can compute many other related parameters such as student overlap, and we are able to compute exactly how a student will travel once some of his selected module is reallocated to the new campus.

A.2: Details on the Surrogate Objective Function

This section describes the algorithm to generate the surrogate objective function and the analysis on the number of variables for this function. These information are tracked back to Section 4.2.2. We first describe the algorithm to generate $\{I_1^k, I_2^k \mid \forall k\}$ which is used to construct the surrogate function:

let $k \in \{1, \dots, |I|\}$ be the index of iterations and $S_k = \{I_1, \dots, I_{|I|-k}\}$ be the set of module-groups in iteration k .

Step 1. $k = 0$. Initialize $S_0 = \{\{1\}, \dots, \{I|\}\}$. Construct $2|T|$ matrixes $\{\Gamma_t^d, \forall d \in \{1, 2\}, t \in T\}$ in the following way:
 $\Gamma_t^d = \left[\Gamma_{(mn)_t}^d, \forall m, n \in \{1, \dots, |I|\} \right], d \in \{1, 2\}, t \in T$ where $\Gamma_{(mn)_t}^d$ is the element in row m and column n of matrix Γ_t^d . If $m < n$, its value equals to $r_{I_m I_n t}^d$; Otherwise, it is 0.

Step 2. Among the $2|T|$ matrixes, find element $\Gamma_{(m^*n^*)_t}^{d^*}$ such that $\Gamma_{(m^*n^*)_t}^{d^*} = \max_{d', t', m', n'} \Gamma_{(m'n')_t'}^{d'}$. Denote $i_1^k = \max(m^*, n^*)$, $i_2^k = \min(m^*, n^*)$. Set $I_1^k = I_{m^*}$ and $I_2^k = I_{n^*}$. Then combine module-set I_{m^*} and I_{n^*} and thus updating S_k . At the same time, update the $2|T|$ matrixes by (1) setting $\Gamma_{(m'm^*)_t}^d = \Gamma_{(m'm^*)_t}^d + \Gamma_{(m'n^*)_t}^d, \forall m' \neq m^* \text{ or } n^*, t \in T, d \in \{1, 2\}$ and (2) deleting row and column n^* in all $2|T|$ matrixes.

Step 3. If $k < |I|$, then $k = k + 1$ and go to step 2, otherwise ends.

In addition, we analyze the scale of variables related to the surrogate measure of traffic.

Theorem 1. The number of auxiliary variables $\{F_{kl}\}$ is $O(|I| \log |I|)$.

Proof:

In iteration k of the aforementioned algorithm, two groups are combined and a group pair $\langle I_1^k, I_2^k \rangle$ such that $|I_1^k| \geq |I_2^k|$ is found. This group

pair is later used to generate $F_{kl} = \frac{1}{|I_1^k| |I_2^k|} \left| \sum_{i \in I_1^k} \left(\sum_{j \in J} x_{ij} - \sum_{j \in J} x_{lj} \right) \right|, \forall l \in I_2^k$.

Therefore, the total number of auxiliary variables is $\sum_k |I_2^k|$.

Define a function $\pi(n): n \geq 2$ as the maximum number of $\sum_k |I_2^k|$

given $|I| = n$. For instance, $\pi(1) = 0$, $\pi(2) = 1$ as the only combination way is

to combine two elements together and $\sum_k |I_2^k| = |I_2^1| = 1$; $\pi(3) = 2$ as the only

combination way is to combine two of three elements first and then combined

with the one left behind. $\sum_k |I_2^k| = |I_2^1| + |I_2^2| = 1 + 1 = 2$. In fact, we

have
$$\pi(n) = \max_{n' \leq n} \{ \pi(n') + \pi(n - n') + \min(n', n - n') \}$$

$= \max_{n' \leq \lfloor \frac{n}{2} \rfloor} \{ \pi(n') + \pi(n - n') + n' \}$. We use induction to prove $\pi(n) \leq \frac{n}{2} \log_2 n$.

It is obvious that $\pi(n) \leq \frac{n}{2} \log_2 n$ when $n=2$ and 3 . For any $3 \leq n < N$,

assuming as the induction hypothesis that $\pi(n) \leq \frac{n}{2} \log_2 n$. Now look at

$\pi(N) = \max_{n' \leq \lfloor \frac{N}{2} \rfloor} \{ \pi(n') + \pi(N - n') + n' \}$. For a given n' , denote $a = n'$ and

$b = N - a$, then $\pi(a) \leq \frac{a}{2} \log_2 a$ and $\pi(b) \leq \frac{b}{2} \log_2 b$. according to Jensen's

inequality, $\frac{a}{a+b} \log_2 a + \frac{b}{a+b} \log_2 b + \frac{2a}{a+b} \leq \log_2 \frac{a^2 + b^2}{a+b} + \frac{2a}{a+b}$. Note that

$$\frac{(a+b)^2}{a^2 + b^2} = 1 + \frac{2ab}{a^2 + b^2} \geq 2^{\frac{2a}{a+b}} \quad \text{as} \quad a \leq b \Rightarrow RHS \leq 1, \quad \text{we have}$$

$$\log_2 \frac{(a+b)^2}{a^2 + b^2} \geq \log_2 2^{\frac{2a}{a+b}} \Rightarrow \log_2 (a+b) \geq \log_2 \frac{a+b}{a^2 + b^2} + \frac{2a}{a+b}. \quad \text{Hence,}$$

$$\frac{a}{a+b} \log_2 a + \frac{b}{a+b} \log_2 b + \frac{2a}{a+b} \leq \log_2 (a+b) \quad \text{and we have proven}$$

$$\frac{a}{2} \log_2 a + \frac{b}{2} \log_2 b + a \leq \frac{a+b}{2} \log_2 (a+b) \text{ for any valid } (a,b).^{49} \text{ Therefore, we}$$

$$\text{have } \pi(N) \leq \max_{a+b=N, a \leq \lfloor \frac{N}{2} \rfloor} \left\{ \frac{a}{2} \log_2 a + \frac{b}{2} \log_2 b + a \right\} \leq \frac{N}{2} \log_2 N. \quad \square$$

A.3 Test Case Generation for Numerical Experiment

We show that (3.2) to (3.6) linearizes (3.1). We first show that linear function

$t_{i_1 i_2}^A$ with constraints (3.3) to (3.6) and non-linear $t_{i_1 i_2}^B$ are equivalent:

⁴⁹ The equality condition is $a=b$ which means $\log_2 n$ needs to be integral.

$$t_{i_1 i_2 t}^A = r_{i_1 i_2 t}^1 v_{i_1} + r_{i_1 i_2 t}^2 v_{i_2},$$

$$t_{i_1 i_2 t}^B = \max\left(r_{i_1 i_2 t}^1 \max(v_{i_1} - v_{i_2}, 0), r_{i_1 i_2 t}^2 \max(v_{i_1} - v_{i_2}, 0)\right)$$

We enumerate the domain of variable pair (v_{i_1}, v_{i_2}) and show the function values of both in Table 6-3.

Table 6-3 The mapping of the non-linear function and its replacement

Domain	Function value of $t_{i_1 i_2 t}^A$	Function value of $t_{i_1 i_2 t}^B$
(0, 0)	$0 (0 \leq v_{i_1 i_2} \leq \frac{1}{2}(1) \Rightarrow v_{i_1 i_2} = 0)$	0
(0, 1)	$0 (-1 \leq v_{i_1 i_2} \leq \frac{1}{2}(0) \Rightarrow v_{i_1 i_2} = 0)$	0
(1, 0)	$r_{i_1 i_2 t}^1 + r_{i_1 i_2 t}^2 (1 \leq v_{i_1 i_2} \leq \frac{1}{2}(2) \Rightarrow v_{i_1 i_2} = 1)$	$\max(r_{i_1 i_2 t}^1, r_{i_1 i_2 t}^2)$
(1, 1)	$0 (0 \leq v_{i_1 i_2} \leq \frac{1}{2}(1) \Rightarrow v_{i_1 i_2} = 0)$	0

Specifically, we use the fact that variable $v_{i_1 i_2}$ is required to be binary in (3.6) when we derive the function value of $t_{i_1 i_2 t}^A$. Also, as for a given module pair (i_1, i_2) and t , at most one of $r_{i_1 i_2 t}^1$ and $r_{i_1 i_2 t}^2$ can be positive because the traffic contribution can only occur in at most one direction. As such, $r_{i_1 i_2 t}^1 + r_{i_1 i_2 t}^2 = \max(r_{i_1 i_2 t}^1, r_{i_1 i_2 t}^2)$. So $t_{i_1 i_2 t}^A$ is equivalent to $t_{i_1 i_2 t}^B$.

Then, since (3.2) is essentially $\min z$ while $z \geq t_{i_1 i_2}^A, \forall t$ and (3.1) is essentially $\min_t \left(t_{i_1 i_2}^B \right)$, it concludes that our linearization is correct.