MODELING AND HEURISTIC SOLUTIONS OF UNIVERSITY TIMETABLING PROBLEMS

ALDY GUNAWAN

NATIONAL UNIVERSITY OF SINGAPORE

2008

MODELING AND HEURISTIC SOLUTIONS OF UNIVERSITY TIMETABLING PROBLEMS

ALDY GUNAWAN
(B.Eng (Ubaya), M.Sc, M. Eng (NUS))

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2008

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisors, Dr Ng Kien Ming and Associate Professor Poh Kim Leng, not only for their invaluable guidance and useful suggestions throughout the study and research process, but also for their care and assistance during the whole period.

My special gratitude is expressed to my beloved wife, Novianty Tandian, my daughter, Natasha Alexandra Gunawan as well as my family for their special support during my study. I would like to thank to all other lecturers and staff members of the Industrial and Systems Engineering Department, National University of Singapore, for the continuous support, and necessary facilities which are very important in carrying out this research.

I also wish to extend my appreciation to my colleagues and friends in the Computing Lab (ISE department) who have helped me and made my life in NUS enjoyable and fruitful.

Aldy Gunawan

December 2008

GLOSSARY

3-SAT	3-Satisfiability
COP	Combinatorial Optimization Problem
CS	Course Scheduling
EP	Examination Problem
GRASP	Greedy Randomized Adaptive Search Procedure
GRASP-SA-TS	Greedy Randomized Adaptive Search Procedure - Simulated Annealing - Tabu Search
LR	Lagrangian Relaxation
LR-SA	Lagrangian Relaxation - Simulated Annealing
QAP	Quadratic Assignment Problem
QAPLIB	Quadratic Assignment Problem Library
QSAP	Quadratic Semi-Assignment Problem
RTT	Restricted Timetable Problem
SA	Simulated Annealing
SA1	Simulated Annealing 1
SA2	Simulated Annealing 2
SA-TS	Simulated Annealing - Tabu Search
TA	Teacher Assignment
TACS	Teacher Assignment - Course Scheduling
TS	Tabu Search

TABLE OF CONTENTS

SUMMARY

Timetabling is the allocation, subject to constraints, of given resources to objects being places in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives. Timetabling problems arise in a wide variety of domains including education, sport, transport and healthcare institutions.

This research mainly focuses on two categories of the educational timetabling problem at the university level: the course and the examination timetabling problems. There are several differences between both problems. In the examination timetabling problem, a number of examinations can often be scheduled into one room or an examination may be split across several classrooms, where else in the course timetabling problem, it is most typically the case that one course has to be scheduled into exactly one classroom.

The course timetabling problem can be further decomposed into five different sub-problems: teacher assignment, class-teacher timetabling, course scheduling, student scheduling and classroom assignment. In this research, we focus on two sub-problems: teacher assignment and course scheduling problems. Most research works in this area only focus on one of the sub-problems of the course timetabling problem, such as the course scheduling problem where it is often assumed that the teacher assignment problem has already been solved earlier before the actual scheduling of courses to time periods.

Motivated by the need to overcome this limitation of only considering one sub-problem, three different mathematical programming models that combine both teacher assignment and course scheduling problems simultaneously are introduced. This combination is known as the Teacher Assignment - Course Scheduling (TACS) problem. The first mathematical model, TACS Model I, is considered as a basic model which accommodates some common requirements.

This model is further extended to two different models, namely, TACS Model II and TACS Model III, by accommodating some additional requirements.

Initially, we solve these mathematical programming models by using ILOG OPL Studio software. However, this software could not provide optimal solutions especially for large scale instances of TACS Model III. A simple improvement heuristic is proposed in order to obtain the solutions. Since the results obtained by a simple improvement heuristic are not good enough, four algorithms based on hybridization of the Simulated Annealing and other methods are proposed to solve the problem. These algorithms are known as Algorithms SA1, SA2, SA-TS and LR-SA. We conclude that LR-SA outperforms other algorithms in terms of solution quality.

This research focuses not only on the course timetabling problem, but also on the examination timetabling problem. In the examination timetabling problem context, majority of the methods proposed were centered on the general concepts of graph theory. However, some constraints, such as students cannot take two examinations consecutively, limit the use of graph theory approach. We formulate the basic examination timetabling problem as a Quadratic Assignment Problem in order to overcome this limitation.

Algorithm GRASP-SA-TS is proposed in order to solve the problem. This hybrid algorithm is based on a combination of Greedy Randomized Adaptive Search Procedure, Simulated Annealing and Tabu Search. The proposed algorithm is able to obtain the optimal or the best known solutions for several QAP benchmark problems.

In real-world situations, the number of examinations can be greater than the number of time periods. There should be a possibility to assign more than one examination to a time period. Therefore, the basic examination timetabling problem is extended to a more general model. One of the constraints in QAP is relaxed and the entire model is formulated as a Quadratic

Semi-Assignment Problem (QSAP). Algorithm GRASP-SA-TS is modified in order to solve the extended examination timetabling problem. The computational results show the ability of the hybrid algorithm to provide good quality solutions compared with those of pure SA.

In summary, this study focuses on the course timetabling and examination timetabling problems. In the course timetabling problems, teacher assignment and course scheduling problems sub-problems are studied simultaneously which is not commonly studied by other researchers. Several algorithms based on hybridization of the Simulated Annealing and others methods are proposed to solve the problem. The idea of the hybrid algorithms is also applied to the examination timetabling problems that have been formulated as a QAP and QSAP in this thesis. A hybrid algorithm based on a combination of GRASP, Simulated Annealing and Tabu Search is introduced to solve the problem.

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

The following nomenclatures will be used throughout the thesis:

i	Index of teacher
j	Index of course
k	Index of course section
l	Index of day
m	Index of time period
I	Set of all teachers
J	Set of all courses
K	Set of all course sections
L	Set of all days available
M	Set of all time periods available
J_i	Set of courses that could be taught by teacher i
K_j	Set of sections of course j
N_i	Maximum number of courses taught by teacher i
H_j	Number of time periods required for course j
V_i	Maximum number of courses taught by teacher i per day
LT_j	Minimum number of teachers that could teach course j
UT_j	Maximum number of teachers that could teach course j
Sec_j	Number of sections of course j
PC_{ij}	Value given by teacher i on the preference of being assigned to teach course j
PT_{im}^{I}	Value given by teacher i on the preference of being assigned to teach at time period m (TACS Model I and II)
PT_{ilm}^{III}	Value given by teacher i on the preference of being assigned to teach on day l and at time period m (TACS Model III)

C_m^{I} Maximum number of classrooms available during time period m (TACS Model I and II)

C_{lm}^{III} Maximum number of classrooms available on day l and at time period m (TACS Model III)

$\lceil a \rceil$ Smallest integer greater than or equal to a

$\lfloor a \rfloor$ Largest integer less than or equal to a

$|A|$ Number of elements in the set A

$\|.\|$ Norm of vector

CHAPTER I

INTRODUCTION

Timetabling is the allocation, subject to constraints, of given resources to objects being places in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives (Wren, 1996). Burke et al. (2004) provide a general definition of timetabling:

> *A timetabling problem is a problem with four parameters: T, a finite set of times; R, a finite set of resources; M, a finite set of meetings; and C, a finite set of constrains. The problem is to assign times and resources to the meetings so as to satisfy the constraints as far as possible.*

Timetabling can be considered to be a certain type of scheduling problem (Petrovic and Burke, 2004). Scheduling is the allocation, subject to constraints, of resources to objects being placed in space-time, in such a way as to minimize the total cost of some set of the resources used (Wren, 1996) Some different views on the terms scheduling and timetabling can be found in the literature. For example, scheduling often aims to minimize the total cost of resources used, while timetabling often tries to achieve the desirable objectives as nearly as possible, such as teacher preferences.

Carter (2001) points out that timetabling decides upon the time when events will take place, but it does not usually involve the allocation of resources in the way that scheduling often does. For example, a published bus or train timetable shows when journey are to be made on a particular routes. It is not necessary to inform which vehicles or drivers are assigned to that particular journey. On the other hand, a university course timetable has also take into account the availability of individual lecturers. The activities of drawing up the university course

timetable maybe considered as a scheduling activity. Some problems may fit more than one of the above definitions, and the terms tend to be used rather loosely in the workplace and in the scheduling community (Wren, 1996; Petrovic and Burke, 2004).

An assignment problem is the problem of assigning a group of individuals to a certain number of jobs (Gass and Harris, 1996). In the process of building a university timetable, an assignment process is involved in certain sub-problems, such as assigning teachers to courses/course sections (teacher assignment problem) and assigning courses to classrooms (classroom assignment problem).

The timetabling problem has attracted substantial research interests due to its importance in a wide variety of application domains, including education (e.g. Burke et al. 2004), transport (e.g. Kwan, 2004), employee/staff (e.g. Schaerf and Meisels, 2000), healthcare institutions (e.g. Bellanti et al., 2004 and Burke et al., 2004) and sport (e.g. Schönberger et al., 2004). The International Series Conferences on the Practice and Theory on Automated Timetabling (PATAT) which is held bi-annually is evident enough for the increase in research activities in this particular area.

In this study, we focus on the educational timetabling problem at the university level. In a general educational timetabling problem, a set of events (e.g. courses and examinations) need to be assigned into a certain number of time periods subject to a set of constraints, which often makes the problem very hard to solve in real-world circumstances.

Timetabling problem is an important problem encountered in every university throughout the world. A very primitive and naïve version of timetabling problem, namely, the restricted timetable problem (RTT), is shown to be NP-complete (Even et al., 1976) and therefore, all the

other variants also lead to NP-complete problem (Karp, 1972). RTT problem is a timetabling problem with the following restrictions: the number of time periods = 3, all classes are always available at each time period, each teacher teach either 2 or 3 classes. The proof of it was shown by displaying a polynomially bounded reduction of the 3-Satisfiability (3-SAT) to RTT. 3-Satisfiability is a special case of *k*-Satisfiability (*k*-SAT) or simply Satisfiablity (SAT), when each clause contains exactly $k = 3$ literals. It was one of NP-complete problems (Karp, 1972). Some other well-known NP-complete problems, such as GRAP K-COLOURABILITY, BIN PACKING and 3-DIMENSIONAL MATCHING, can be reducible to the timetabling problem (Cooper and Kingston, 1996).

Educational timetabling problem is mainly classified into two different categories: course timetabling and examination timetabling problems. The course timetabling problem is defined as a multi-dimensional scheduling problem in which students, teachers are assigned to courses, course sections or classes; "events" (individual meetings between students and teachers) are assigned to classrooms and times (Carter and Laporte, 1998). The examination timetabling problem can be defined as the problem of allocating a number of examinations to a certain number of time periods in such a way that there would be no conflict or clash, i.e., no student are required to attend more than one examination at the same time period (Carter and Laporte, 1996).

Burke and Petrovic (2002) highlighted several similarities and differences of both problems. The course timetabling problem can be further decomposed into five different sub-problems: teacher assignment, class-teacher timetabling, course scheduling, student scheduling and classroom assignment (Carter and Laporte, 1998). The details would be explained in Section 1.2.

In this introductory chapter, the background information, scope of the study, objectives of this study as well as the organization of this thesis are presented.

1.1 Background and Motivation

As mentioned above, one of the key applications of timetabling is in the matter of educational timetabling. Three significant developments that increase the interest in these problems are (de Werra, 1985 and Johnson 1993):

a. The huge variety of problems faced due to different requirements in each institution. Schaerf (1999) gives a survey of the various requirements and formulations of timetabling problems.
b. The nature of education. In particular, the timetable has become much more complicated due to changes in the educational systems, such as new subjects introduced, facility requirements, number of students and teachers involved, a much greater range of choice in the subjects that students can take; hence the problem needs a regular modification to adapt to the requirements of a changing environment.
c. Computing facilities and expertise are now available in most education institutions. Computer and database system are widely used because they could provide high-level information storage and processing. Computer is able to cope the complexity, the changes, such as introduction of new courses (McCollum, 1998).

Educational timetabling problem is one of the most important and time consuming tasks which occurs periodically in all institutions around the world. During the last few decades, many contributions related to timetabling problems have appeared. Several approaches or methods have been proposed for solving these problems in the literature (e.g. de Werra, 1985; Carter and Laporte, 1998; Schaerf, 1999 and Burke and Petrovic, 2002).

As mentioned earlier, educational timetabling problem is mainly classified into two different categories: course timetabling and examination timetabling problems. In the course timetabling context, the primary problem faced is to schedule asset of teachers to courses within a given number of rooms and time periods. The course timetabling problem can be further decomposed into five different sub-problems: teacher assignment, class-teacher timetabling, course scheduling, student scheduling and classroom assignment.

In the course timetabling context, we focus on two sub-problems: teacher assignment and course scheduling problems. We notice that many papers only focus on one of the sub-problems, as in the works of Andrew and Collins (1971), Harwood and Lawless (1975), Tillett (1975), Breslaw (1976), Schniederjans and Kim (1987) and Wang (2002) that only focus on the teacher assignment problem.

On the other hand, the course scheduling problem only focuses on allocating courses to time periods. It is often assumed that the allocation of teachers to courses has been done earlier before the actual scheduling of courses to time periods, as in the works of Daskalaki et al. (2004), Al-Yakoob and Sherali (2006, 2007) and Lewis and Paechter (2007). This limitation motivates us to solve both problems simultaneously since real life problems always contain a combination of some of the sub-problems.

In the examination timetabling context, majority of the methods proposed were centered on the general concepts of graph theory or network analysis. However, some constraints, such as students cannot take two examinations consecutively, limit the use of graph theory approach (Lewis, 2008). We formulate the basic examination timetabling problem as a Quadratic Assignment Problem in order to overcome this limitation. QAP formulation can be used to deal with conflicts in the examination timetabling problem, such as no students can take two or

more examinations at the same time, room capacities as well as several other constraints (Bullnheimer, 1997).

1.2 Scope of the Study

As described in Section 1.1, the educational timetabling problem can be classified into two main categories: course timetabling and examination timetabling problems (Burke, 2002) (Figure 1.1). Another type of classification was proposed by Scaherf (1999). The educational timetabling problem is categorized into three main categories:

- School timetabling: The weekly scheduling for all the classes of a school with the purpose to avoid teachers meet two classes at the same time, and vice versa.
- Course timetabling: The weekly scheduling for all the lecturers of a set of university courses in order to minimize the overlaps of teachers having common students.
- Examination timetabling: The scheduling for the examinations in order to avoid overlap of examinations having common students and to spread the examinations for the students as much as possible.

In this research, we refer to the classification of timetabling presented by Burke (2002). The scope of this study covers both course timetabling and examination timetabling problems at the university level. Although each category is studied separately in this research, in fact, they can be combined to make a comprehensive/complete analysis with the objective of achieving further improvement in the university timetabling area.

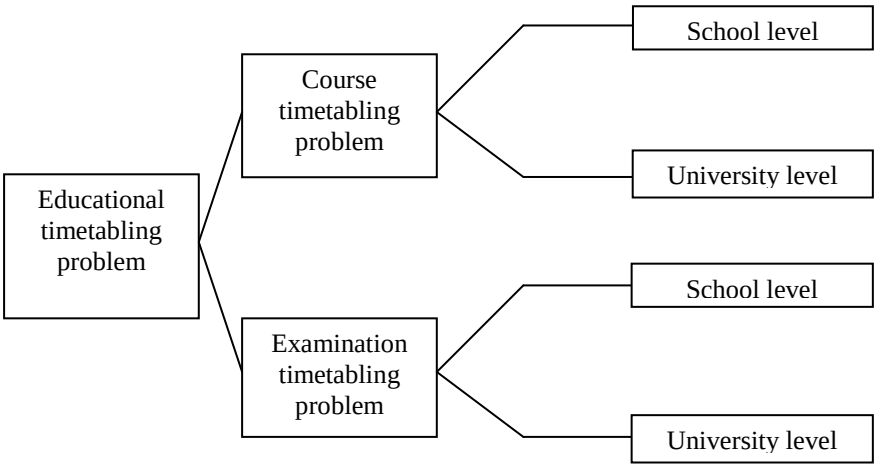


Figure 1.1 Classification of educational timetabling problems

In the course timetabling context, there are several differences between university and school timetabling problems. In school timetabling problem, we often work with predefined classes and schools have few programs. In universities, more programs are offered and faculty members/teachers may only teach few hours a week.

In the examination timetabling context, universities schedule examinations in order to avoid overlap of examinations having common students and to spread the examinations for the students as much as possible since students might take different courses. On the other hand, the program at the school level is usually highly structured and very tight. Students who take similar courses are divided into several classes. The examination timetabling problem focuses on how to schedule the examinations for each class. It is common that each class might be required to take two examinations consecutively.

1.3 Purpose of the Study

The proposed research mainly focuses on the course timetabling and the examination timetabling problems at the university level. The overall objective is to solve both problems by

proposing several methods and then perform a comparative study of the proposed methods based on the computational results obtained. Detailed discussion about each method would be included in order to facilitate better understanding of the problems.

Specifically, this study focuses on several topics regarding the university timetabling problems in order to fulfill the following targets.

- To identify existing limitations of the university timetabling problems.
- To study the university course timetabling problems by considering two sub-problems, namely, teacher assignment and course scheduling problems simultaneously, known as the Teacher Assignment - Course Scheduling problem (TACS problem).
- To develop new mathematical programming models for the TACS problem.
- To propose and compare several algorithms, including hybrid algorithms, for solving the proposed mathematical programming models.
- To present the examination timetabling problem as a Quadratic Semi-Assignment Problem.
- To propose hybrid algorithms for solving the examination timetabling problem.

1.4 Organization of the Thesis

This thesis consists of eight chapters. Chapter 1 introduces the problem along with the necessary background and motivation for the problem. The chapter also details the scope and the purpose of the undertaken study. Chapters 2 to 8 elaborate on the different problems studied in the context of university timetable scheduling.

Chapter 2 presents a thorough and comprehensive literature review of the educational timetabling studies in the recent years. The details of university timetabling classification, the theory of the timetabling problems, including formulations of the timetabling problem as well

as summary of the algorithms that have been applied to the educational timetabling problems, are described. In addition, a brief overview of hybrid algorithms is summarized.

In Chapter 3, a detailed description of the TACS problem is presented. The basic and extended TACS mathematical models are introduced. Computational experiments based on some randomly generated instances are summarized and discussed in detail. The limitation of commercial software used for solving the TACS problem is highlighted.

This situation inspires us to propose an improvement heuristic in order to solve the problem, which is further discussed in Chapter 4. The heuristic applies the principles of a simple greedy heuristic. Finally, the computational results obtained are presented and analyzed. Based on the results obtained, further improvements of the proposed heuristic are introduced in Chapters 5 and 6. Several hybrid algorithms are proposed and compared comprehensively, including the computation time and the solution quality. The algorithms apply the principles of two well-known metaheuristics, Simulated Annealing and Tabu Search. Finally, conclusions of the performances of the proposed algorithms are presented.

In Chapter 7, the examination timetabling problem is studied. The basic and extension models of this problem are presented. This chapter extends the idea of the proposed hybrid algorithm presented in the previous chapters to the examination timetabling problem. Computational experience on a set of standard test problems (Quadratic Assignment Problem Library - QAPLIB) and several random data sets are summarized. The results obtained are compared with the best known/optimal solutions or lower bound of the problems. At the end, some conclusions, major contributions of the study, as well as limitations and suggestions for future research are presented in Chapter 8.

CHAPTER II
LITERATURE REVIEW

2.1 Introduction

Many optimization problems concern with the choice of the best configuration of a set of variables to achieve some goals. Combinatorial optimization problem is considered as a class of problems where the set of feasible solutions is discrete (Blum and Roli, 2003).

Definition 2.1 (Blum and Roli, 2003) A Combinatorial Optimization problem $P = (S, f)$ can be defined by:

- A set of variables $X = \{x_1, \ldots, x_n\}$;
- Variable domains $D_1, \ldots, D_n$;
- Constraints among variables;
- An objective function f to be minimized or maximized, where f: $D_1 \times \ldots \times D_n \rightarrow R^+$;

The set of all possible feasible assignments is

$$S = \{s = \{(x_1,v_1),\ldots, (x_n,v_n)\} | v_i \in D_i, s \text{ satisfies all the constraints}\}$$

Timetabling problems belong to a class of combinatorial optimization problems (COPs). A survey of related applications and approaches of combinatorial optimization was given by Grötschel (1991). Timetabling problem is defined as the problem of assigning a number of events into limited number of time periods. In this research, the focus would be concentrated on university timetabling problems. Timetabling problems are numerous which every university may have different characteristics due to different types of constraints or requirements.

Timetabling problems have attracted the attention of the Operation Research and Artificial Intelligence community. For surveys of timetabling methods and applications see de Werra (1985), Carter and Laporte (1998), Schaerf (1999) and Burke and Petrovic (2002).

In this chapter, a detail description about educational timetabling problem especially in the university timetabling problem will be presented. We also present a brief overview of the hybrid algorithm including its applications in the timetabling problem.

2.2 The Classification of the Timetabling Problem

Educational timetabling problem can be divided into two main different categories: course and examination timetabling problems (Burke and Petrovic, 2002). The main differences between course timetabling and examination timetabling problems are summarized in the following table. Each category would be described in the following sub-sections.

Table 2.1 Differences between course and examination timetabling problems

Course Timetabling	Examination Timetabling
One room can only be used for one course	Several exams can be done in one room or an exam might be split across several rooms
Students may have two or more courses in a adjacent time periods	Students may not have too many consecutive exams

2.2.1 The Course Timetabling Problem

The purposes of course timetabling are either to assign students, teachers to courses, course sections or classes or to assign courses, course sections or classes to time periods and/or classroom or both. The course timetabling problem can be viewed as a multi-dimensional scheduling problem in which students, teachers are assigned to courses/course sections;

meetings between students and teachers are assigned to classrooms and times. The course timetabling problems are applied to both school and university levels (Figure 1.1).

Carter and Laporte (1998) presented the major differences between course timetabling problem at school and university levels, as shown in Table 2.2. Although the course timetabling problem structure varies among institutions, several common components, such as definitions of course, class and program were presented by Carter and Laporte (1998).

Table 2.2 Differences between school and university course timetabling problems

Characteristic	School	University
Scheduling	- By classes	- By students
Choice	- Only few choices	- Many electives
	- Highly structured programs	- Loosely structured programs
Teacher availability	- Heavy teaching load	- Light teaching load
Rooms	- Only few rooms used	- Many rooms used
	- Same size	- Variety of sizes
Student load	- Very heavy	- Fairly light
Criteria	- No conflicts	- Minimum conflicts
Available rooms	- Negligible	- Limited

Based on the planning sequence of timetable arrangement, Carter and Laporte (1998) distinguished the course timetabling problems as master timetable system and demand driven system. In the master timetable system, the institution releases the course timetable (including their sections and times) and students choose courses from the published timetable based on their preferences. In the latter, the institution releases only the course offered. Students select their courses from the list. The number of sections and time periods will then be decided based on the student requirements.

Some examples of master timetable systems are course timetabling systems at the Canadian Engineering School (Laporte and Desroches, 1986), the Anderson School of Management

UCLA (Stallaert, 1997), the University of Nottingham (McCollum, 1998) and the Syracuse University (Saleh Elmohamed et al., 1998). It was highlighted that it would be unworkable in practice, if the institution allows the students' choice to dictate the timetable. On the other hand, some universities have applied the demand driven system, for instance, the Darden Graduate School (Sampson et al., 1995), the University of Valencia Spain (Valdes et al., 2000) and the University of Waterloo (Carter, 2001).

Carter and Laporte (1998) decomposed the course timetabling problem into five different sub-problems: teacher assignment, class-teacher timetabling, course scheduling, student scheduling, and classroom assignment. The real life timetabling problems always contain a combination of the sub-problems (Figure 2.1), although not all of sub-problems may be relevant to a particular situation For example, the university course timetabling problem may consist of four sub-problems: teacher assignment, course scheduling, student scheduling and classroom assignment.

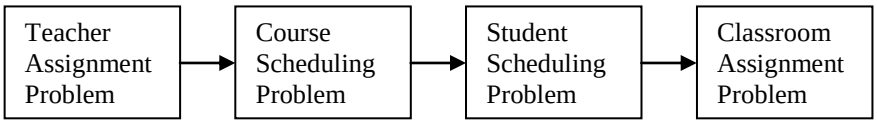


Figure 2.1 University course timetabling problem

Each sub-problem focuses on a different problem, for instance, course scheduling problem only focuses on allocating courses to time periods by assuming the information about which teachers will be allocated to which particular course has been decided. The following figures illustrate the difference between the course timetabling problem and the course scheduling problem.

Day	1					2				
Period	1	2	3	4	5	1	2	3	4	5
Teacher1								4(2)	4(2)	
Teacher2	3(1)*	3(1)								
Teacher3		5(1)	5(1)							
Teacher4						2(2)	2(2)			
Teacher5						1(2)	1(2)			

*Course 3 Section 1 taught by Teacher2 is scheduled on Day 1 Time periods 1 and 2

Figure 2.2 A numerical example for the course scheduling problem

Day	1					2				
Period	1	2	3	4	5	1	2	3	4	5
Teacher1								4(2) R2	4(2) R2	
Teacher2	3(1) R1*	3(1) R1								
Teacher3		5(1) R2	5(1) R2							
Teacher4						2(2) R3	2(2) R3			
Teacher5						1(2) R1	1(2) R1			

*Course 3 Section 1 taught by Teacher2 is scheduled on Day 1 Time periods 1 and 2 in Room 1

Figure 2.3 A numerical example for the course timetabling problem

Each sub-problem will be described below:

- Teacher Assignment Problem

The aim is to assign teachers to the courses by considering their preferences. Some researchers have discussed the teacher assignment problem in the literature. One of the earliest papers was written by Andrew and Collins (1971). The problem is about how to make teacher assignments that have high effectiveness and preference ratings, while ensuring that all courses will be staffed and no teacher is overloaded. Tillett (1975) argued that that model could not be applied in the secondary school context. The extended model has been developed by considering preparation factor.

Most of the early research work did not consider conflicting goals in the assignment problem. The natural conflict between competing individual teachers in course-teacher assignment can be represented as a goal programming model (Schniederjans and Kim, 1987). Some factors

that could affect the size and complexity of the teacher assignment problems were presented. They did mention that predetermined assignments could reduce the assignment problem size and complexity. However, such models usually do not deal with the course scheduling problem.

Badri (1996) proposed a two-stage optimization procedure, this model seeks to maximize teacher-course preferences in assigning teachers to courses, and then maximize teacher-time preferences in allocating courses to time periods. Wang (2002) applied Genetic Algorithm for solving teacher assignment problem at Far East College, Taiwan.

- Course Scheduling Problem

The aim is to assign courses or course sections to time periods provided. Course scheduling can be considered as the most discussed problem in recent years, as witnessed by the work of Aubin and Ferland (1989), Abramson (1991), Hertz (1991, 1992), Abramson et al. (1999) and Gunawan et al. (2004). This problem can be combined with another sub-problem, classroom assignment problem. For more details, see Saleh Elmohamed et al. (1998), White and Zhang (1998), Daskalaki et al. (2004) and Al-Yakoob and Sherali (2006, 2007). However, it is often assumed that the teacher assignment has been solved and fixed before solving the course scheduling problem (Stallaert, 1997; Daskalaki et al., 2004; Daskalaki and Birbas, 2005; Al-Yakoob and Sherali, 2006 and 2007).

- Class-Teacher Timetabling Problem

Students who take a similar group of courses are arranged into a class. Here, the scheduling unit is a class, not a student. This problem mostly arises in the school level. The main purpose is to construct a schedule for class-teacher meetings. It is assumed that the assignment of teachers to courses and classes has been determined. Class-teacher timetabling problem

without side constraints can be solved in polynomial time by means of a network flow algorithm (de Werra, 1971). Several papers discusses about this problem were published in recent years (Chalal and de Werra, 1989; Abramson, 1991; Costa, 1994; Schaerf, 1999). Asratian and de Werra (2002) presented a generalized class-teacher model which extends the basic class-teacher model.

- Student Scheduling Problem

The main purpose in this scheduling is to assign the students to the course section while balancing section sizes and respecting room capacities. This problem occurs especially when courses are taught in multiple sections (Carter and Laporte, 1998). Once students have selected their courses, they must be assigned to sections. This process can only be done after the university publishes the timetable and students register the courses that they are willing to take. Some papers discuss about this problem were written by Laporte and Desroches (1986), Sabin and Winter (1986) and Graves et al. (1993).

- Classroom Assignment Problem

Courses have to be assigned to specific rooms and time periods. For simplification, the assignment of courses to the time periods is usually done before the assignment of courses to the rooms (Glassey and Mizrach, 1986; Gosselin and Truchon, 1986 and Carter, 1989). Some classroom assignment problems can be considered as easy problems although others may be difficult to solve (Carter and Tovey, 1992). Both the non-interval and interval classroom assignments are proven as NP-complete based on reduction from 3-SAT (3-Satisfiability). 3-SAT is a special case of *k*-satisfiability (*k*-SAT) when each clause contains exactly $k = 3$ literals (Garey and Johnson, 1979). For fixed number of periods, the non-interval problem can be solved in polynomial time.

2.2.2 The Examination Timetabling Problem

The examination timetabling problem is defined as assigning a set of examinations to a limited number of time periods in such a way that no student can take more than one examination at any time period as well as several other constraints. Conflicts in the examination timetabling problem can be divided into three different categories: the first order conflicts, second order conflicts and higher order conflicts (Leong and Yeong, 1987 and Bullnheimer, 1998).

The first order conflicts refer to a situation where a student has to take two or more examinations at a time period. The second order conflicts term a situation where a student has to take two consecutive examinations. Finally, there may be further constraints dealing with room capacities, pre-scheduled examinations that so-called higher order conflicts.

Carter (1986) presented a review of the early research on practical applications of examination timetabling in several universities. Algorithms implemented in the examination timetabling problem were summarized by Carter and Laporte (1996). They are categorized into four different types: cluster methods, sequential methods, generalized search (metaheuristics) and constraint based techniques that would be explained in Section 2.4.2. A comprehensive survey of British universities was presented by Burke et al. (1996). It covers the structure of the examination problems faced by universities, the ways to solve the problems as well as the objective of the examination timetabling problem.

The examination timetabling problem was also summarized in the review papers of Schaerf (1999), Dimopoulou and Miliotis (2001) and Burke and Petrovic (2002). Burke and Petrovic (2002) gave an overall review of recent research conducted on course and examination timetabling in the university level.

One of the latest reviews is written by Qu et al. (2006). They highlight the search methodologies, automated approaches and the new trends for the examination timetabling problem as well. A range of relevant important research issues and research achievements that have been carried out in the last decade were presented.

2.3 Formulation of the Timetabling Problem

Timetabling problems are subject to several requirements or constraints that are usually classified into two different types: hard and soft constraints (Burke et al., 1996 and Burke and Petrovic, 2002). Hard constraints are rigidly enforced by the institution and, therefore, have to be satisfied. A feasible timetable is one that satisfies all the hard constraints which are commonly embodied as constraints in the mathematical formulation (Costa, 1994).

Soft constraints are those that it is desirable to satisfy, but they are not essential. In real-world timetabling problem, it is usually impossible to satisfy all of the soft constraints. In general, soft constraints are often stated as penalties in the objective function that need to be minimized. The determination and classification of the soft constraints vary extensively in different universities depending on their specific requirements.

Table 2.3 represents a comprehensive list of several common hard and soft constraints for the course timetabling (Burke and Petrovic, 2002) and examination timetabling problems (Qu et al., 2006).

Table 2.3 Primary Constraints in the timetabling problem

Constraint	Course Timetabling Problem	Examination Timetabling Problem
Hard Constraints	− No student and teacher attend more than one course at any time period − the number of classrooms available is restricted at any time period	− no student takes more than one examination at any time period − Resource of examinations (the number of classrooms and their capacity) need to be sufficient
Soft Constraints	− Some courses may need to be scheduled in certain particular time periods − One course may need to be scheduled before/after the other − Teachers might request certain time periods and prefer to teach in a particular classroom	− Some examinations require specific time periods or specific classrooms − Certain examinations are required to be in consecutive time periods − Certain examinations are required to be on the same day − Students should not take two or more consecutive examinations

Several different types of constraints classifications were also proposed by several other researchers. Birbas et al. (1997) classified the constraints based on the feasibility and quality criteria. All the feasibility rules are related to hard constraints. Costa (1994) used different terms to classify the constraints. They were divided into two partitions: essential and relaxed constraints.

Some basic and traditional models of the timetabling problem with several variations were presented as bipartite multigraphs by de Werra (1985). The author summarized some basic class-teacher problems with several variations like pre-assignment schedules for several teachers or classes and unavailability schedules for several teachers or classes. All the problems were presented as integer programming models and the graph colouring models. Indeed, when other real world constraints considered in a problem (particularly those relating

to the ordering of events within a timetable), then the simple graph coloring model will not be sufficient on its own (Lewis, 2008).

The integer programming modeling were widely used for formulating other sub-problems: student assignment problem (Valdes et al., 2000), course scheduling problem (Birbas et al., 1997 and Valdes et al., 2002) and teacher assignment problem (McClure and Wells, 1984 and Wang, 2002). However, Yu and Sung (2002) argued that group coloring algorithm could not incorporate the non academic constraints into the problem formulation. They also mentioned that the integer programming approach would encounter some modeling difficulties when the number of variables and constraints increase.

Other types of formulations are proposed by several researchers. Gosselin and Truchon (1986) formulated the classroom allocation problem as a linear programming model. Aubin and Ferland (1989) and Hertz (1991) formulated the large scale timetabling problem as an assignment problem. Tripathy (1992) presented the course scheduling problem as a modification of the transportation problem with the addition of conflict matrix constraints.

The basic examination timetabling problem is commonly modeled as a graph colouring problem (White and Chan, 1979 and Bullnheimer, 1998). The role of graph colouring methods in the timetabling literature was highlighted in Burke et al. (2004). The Quadratic Assignment Problem has been used to formulate the examination timetabling problems, as in the works of Leong and Yeong (1987). Bullnheimer (1998) formulated the basic examination scheduling problem as a QAP with a different objective function. A fuzzy set based approach has been used to modeling constraints imposed on university examination timetabling problem (Petrovic et al., 2005).

2.4 Algorithms for the Timetabling Problem

There are a significant number of techniques or approaches to timetabling problems that have appeared in the literature. Many applications of the various approaches for solving the problem have been extensively studied and published in Operations Research literature over the last decades. A wide variety of approaches to timetabling problems have been described in the literature. These approaches can be classified with respect to different criteria.

Here, we summarize the classification of algorithms for the course timetabling problem presented by Carter and Laporte (1998). Carter and Laporte (1996) also published a survey of papers on practical examination timetabling problems, including the classification of the algorithms.

2.4.1 Algorithms for the Course Timetabling Problem

Carter and Laporte (1998) classified the algorithms used to solve course timetabling problems into four different groups: global algorithms, constructive heuristics, improvement heuristics and interactive systems.

2.4.1.1 Global Algorithms

Small size problems can sometimes be solved by means of a standard integer linear programming package, such as CPLEX. However, when problem size increases rapidly, the optimal solution could not be found due to computational intractability.

Research has been done on the applications of global algorithms to timetabling problems. Andrew and Collins (1971) developed a procedure based on a simple linear programming technique for assigning the teachers to courses. Some drawbacks of the proposed model were highlighted by Tillet (1975). He noted that the model does not consider some factors such as the number of courses taught by each teacher.

Tillet (1975) proposed an integer programming model for the teacher assignment problem in the secondary school level and solved the model by using commercial software. Breslaw (1976) highlighted the major drawback of the proposed model by Tillet (1975), namely, that the computation time of solving the model can be prohibitively large as the problem size increases. He proposed a model to overcome this major drawback and could be applicable at the university level.

Tripathy (1980) developed an algorithm based on the Lagrangian relaxation approach to course timetabling. The aim was to determine the number of periods required to schedule the courses. The results obtained were compared with Barham and Westwood's results (1978). The author argued that the algorithm could produce the guaranteed minimum objective value.

Tripathy (1992) employed a Lagrangian relaxation approach coupled with sub-gradient optimization for solving a large university timetabling problem. This approach was incorporated with a branch and bound procedure for further improvement. This study has shown not only the potential of the proposed technique for solving large scale timetabling problems but also possible extensions to other similar problems. Carter (1989) used a Lagrangian relaxation technique to solve the classroom assignment problem.

The timetable problem for Greek high schools was formulated by Birbas et al. (1997) as an integer programming model. The paper focuses on creating schedules for schools with major and elective courses and several student groups. CPLEX solver was used to obtain the optimal solution.

The use of linear programming to assign classrooms on a daily basis has been adopted by Gosselin and Truchon (1986). They proposed two stages for solving the classroom allocation. Due to cost and preparation reasons, the rooms and requests were grouped into categories and types, respectively. The first step is to use a simplex algorithm for finding how many requests of each type should be met with rooms of each category. To ensure that the problem always has a feasible solution, fictitious rooms were introduced. In the second step, a simple algorithm is proposed in order to assign a particular room to each request. Those assignments have to be compatible with the solution generated from the first step.

Dimopoulou and Miliotis (2001) developed a system that produces a combined course and examination timetable schedule for the Athens University of Economics and Business. In the course scheduling part, an integer programming model was used. The examination timetabling was developed by using a simple heuristic method.

Some of the latest applications of integer programming are presented by Daskalaki et al. (2004), Al-Yakoob et al. (2006, 2007). They modeled university timetabling problems as integer programming models. Different sizes of problems have been solved by the commercial IP solver software, CPLEX.

Harwood and Lawless (1975) used the goal programming model to formulate the teacher assignment problem. Schniederjans and Kim (1987) highlighted the major drawback of the

Harwood and Lawless' model. The model might be very difficult to implement. They presented some factors that could affect the size and complexity of the teacher assignment problem and proposed a model to overcome the major drawback.

Badri (1996) proposed a goal programming model for solving both course scheduling and teacher assignment problems through a two-stage optimization procedure. The model seeks to maximize teacher-course preferences in assigning teachers to courses, and then maximize teacher-time preferences in allocating courses to time periods. The extended version was proposed by Badri et al. (1998). The model was built through a one-stage process in which both preferences are considered simultaneously.

2.4.1.2 Constructive Heuristics

When the problem size gets larger, it would be difficult to find and prove the existence of an optimal solution, especially within a short computation time (Costa, 1994). It would be necessary to develop a heuristic approach in order to find a good solution within a reasonable amount of time.

The heuristic method tries to find a feasible solution by making sequential assignments. However, backtracking might be needed in order to undo some of the previous allocations. The heuristic method usually consists of two phases (Eiselt and Laporte, 1987). In phase I, we try to find a feasible initial solution, and in phase II, the current feasible solution is improved in order to reach the best optimum.

The large scale course timetabling problems have been solved by several researchers. This problem involves two sub-problems which are related to each other: the course scheduling and

the student grouping problems. Aubin and Ferland (1989) proposed a heuristic approach, namely an iterative descent method, in order to handle both sub-problems iteratively and reach an improved solution.

Wright (1996) proposed a heuristic search for course scheduling in a complex secondary school in Lancashire, England. The courses which have to be simultaneously conducted are grouped into a block. The length of the timetabling period is based on a fortnightly cycle. Both situations cause the course scheduling problem to become more complicated.

Laporte and Desroches (1986) developed a heuristic method in order to allocate students to course sections in Ecole Polytechnique de Montreal (EPM), one of Canada's leading Engineering schools. Abdennadher and Marte (2000) used constraint logic programming as a constructive heuristic for modeling the university course timetabling problem. They showed how to model the timetabling problem as a partial constraint satisfaction problem.

2.4.1.3 Improvement Heuristics

Recently, modern search methods called metaheuristics have been widely used in the timetabling problem. The term metaheuristic was initially introduced by Glover (1986). Some definitions of metaheuristics are given below:

Definition 2.2 A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions (Osman and Laporte, 1996).

Definition 2.3 A metaheuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method (Voss et al., 1999).

Blum and Roli (2003) differentiated metaheuristics into several classifications based on the characteristics: nature-inspired vs. non-nature inspired, population based vs. single point search, dynamic vs. static objective function, one vs. various neighborhood structures, memory usage vs. memory-less methods. The properties of metaheuristics were also pointed out.

The class of metaheuristics includes – but is not restricted to – Simulated Annealing, Tabu Search, Genetic Algorithm, Greedy Randomized Adaptive Search Procedure and Ant Colony Optimization. The latest survey of the application of metaheuristics in the university timetabling problem is presented by Lewis (2008).

SA was initially introduced by Kirkpatrick et al. (1983) for finding good solutions to a wide variety of combinatorial optimization problems. It was one of the first algorithms that had an explicit strategy to avoid local minima. SA has been successfully applied to a variety of combinatorial optimization problems, such as the Traveling Salesman Problem (Cerny, 1985), machine scheduling problem (Van Laarhoven et al., 1992 and Radhakrishnan and Ventura, 2000), the Quadratic Assignment Problem (Wilhelm and Ward, 1987 and Connolly, 1990) and timetabling problem (Abramson, 1991; Thompson and Dowsland, 1996; Bullnheimer, 1998; Abramson et al., 1999; Saleh Elmohamed et al., 1998 and Gunawan, 2004).

The fundamental concepts of Tabu Search (TS) algorithm was originally proposed by Glover (1989, 1990). Many different papers have presented applications of Tabu Search to various combinatorial problems, such as the Quadratic Assignment Problem (Taillard, 1991 and Drezner, 2002 and 2005), machine scheduling problem (Nowicki and Smutnicki, 1996), vehicle routing problem (Gendreau et al., 1998 and 2001) and timetabling problem (Hertz, 1991, 1992 and Costa, 1994).

Hertz (1991) dealt with the large scale course and examination timetabling problems by proposing a new global approach based on TS. Two algorithms proposed are TATI (TAbu search for TImetabling) and TAG (TAbu search for Grouping). The advantage of those techniques compared with an iterative descent method (Aubin and Ferland, 1989) is the ability to avoid being trapped by local optima.

Hertz (1992) proposed a new global approach which is based on the TS. Each subject is divided into several topics. A certain number of time periods have to be assigned to a particular topic. Those time periods should be divided into daily amounts of consecutive time periods called the daily quantum. The fundamental difference with other course scheduling is that the length and the number of some courses is not known in advance.

Some other applications of TS were presented by Costa (1994) for class-teacher timetabling problem, Gunawan et al. (2004) for teacher assignment and course scheduling problems, Valdes et al. (1996) and Valdes et al. (2002) for class timetabling problem and Valdes et al. (2000) for student scheduling problem.

The applications of Genetic Algorithm in the class/teacher timetabling problem were presented by Carrasco and Pato (2001). They used a special multi objective GA, namely the non-

dominated sorting Genetic Algorithm (NSGA). The multi objective GA incorporates two distinct and competitive objectives: teacher-oriented objective and class-oriented objective. The basic idea is to minimize the penalties due to violations of constraints with respect to the two competing aspects: classes and teachers.

The GA was also used by other researchers: Ueda et al. (2001), Yu and Sung (2002) and Gunawan et al. (2004) for course scheduling problem; Wang (2002) and Gunawan et al. (2008) for teacher assignment problem.

The application of Ant Colony Optimization (ACO) in the timetabling problem is not so widely used yet. Socha et al. (2002 and 2003) presented two different ACO algorithms, namely, Ant Colony System (ACS) and MAX-MIN Ant System (MMAS), to solve the simplified version of a typical university course timetabling problem. The data was generated using a generator written by Paechter.

Socha et al. (2002) presented the application of MAX-MIN Ant System (MMAS) combined with the local search routine. The comparison of the MMAS algorithm and a Random Restart Local Search (RRLS) algorithm was presented. It was concluded that the results of MMAS are better than those of RRLS. Socha et al. (2003) compared ACS and MMAS algorithms against some other algorithms such as Simulated Annealing (SA), Iterated Local Search, and RRLS. The MMAS performs better than ACS on all instances tested. While, on medium instances of the problem, SA is better than MMAS, while on the large instances, MMAS has the best performance.

2.4.1.4 Interactive Systems

Some researchers propose interactive (semi–automatic) procedures to handle timetabling problems. Human intervention is still needed especially in the final output (Schaerf, 1999). Timetabling system could be classified by the degree of user and system interaction. Almost all of the current literature describe that the relevant data is entered into the system before the execution starts. It is referred to as the batch oriented system. Only few literatures have used the interactive mode which the updating data could be introduced without having to redo the entire program.

White and Wong (1988) proposed an algorithm that uses piecewise incremental construction. The algorithm is based on the approach presented by Selim (1982). Chalal and de Werra (1989) proposed an interactive system for small class-teacher timetabling. The problem was formulated as a network flow model. The difference with other class-teacher problems was that the authors did not determine who would teach which topic to each class. They concluded that an interactive system is an effective tool.

Carter (2001) presented and summarized a comprehensive interactive course timetabling and student scheduling system developed from 1979 until 1987 at the University of Waterloo. The system developed integrates course timetabling across the institution.

2.4.2 Algorithms for the Examination Timetabling Problem

Algorithms used for solving the examination timetabling problem are classified into four types of approaches: cluster methods, sequential methods, generalized search strategies (metaheuristics) and constraint based approaches (Carter and Laporte, 1996).

2.4.2.1 Cluster Methods

In these methods, examinations with few or without conflicts are grouped into the same block, followed by allocating the blocks into specific periods to satisfy some constraints. These approaches represent a form of decomposition of the problem. Two drawbacks of these methods, infeasibility issue and poor quality solutions, were discussed by Qu et al. (2006).

Leong and Yeong (1990) created conflict free groups by selecting examinations one at a time in descending order of number of conflicts. Other approaches for sorting the examinations are based on the degree times the number of student conflicts with all other examinations (Lotfi and Cerveny, 1991), a linear combination of the size of the enrolment and the number of conflicts with other courses (Johnson, 1990).

2.4.2.2 Sequential Methods

In sequential methods, examinations are assigned to a specific period one at a time, based on some ordering strategies. As explained in Section 2.3, examination timetabling problems can be modeled as graph colouring problems. There is a range of ordering strategies of graph coloring problems can be modified and applied in examination timetabling problems. The following table summarizes some of widely studied ordering strategies in examination timetabling problems (Qu et al., 2006).

Five ordering strategies in Table 2.4 were studied on real and randomly generated examination timetabling problems by Carter et al. (1996). It was concluded that there was no heuristic outperformed any of the rest over the problems tested. Burke et al. (1998) proposed simple techniques in order to improve the solutions by introducing a random element into the

application of heuristics shown in Table 2.4. Asmuni et al. (2005) applied fuzzy logic to arrange the examinations to be scheduled based on ordering strategies on the same data sets.

Table 2.4 Ordering strategies in the examination timetabling problem

No	Heuristics	Ordering strategies
1	Largest degree	Decreasingly by the number of conflicts the examinations have with the other examinations
2	Saturation degree	Increasingly by the number of time periods available for the examination at the time
3	Largest weighted degree	Decreasingly by total number of students in conflict with other examinations
4	Random ordering	Randomly order the examinations
5	Largest enrolment	Decreasingly by the number of enrolments (students) for the examination

2.4.2.3 Generalized Search Strategies (Improvement Heuristics)

These strategies are similar to the improvement heurisctics described in Section 2.4.1.3. Generalized search strategies start with one or more initial solutions and apply a search strategy in order to avoid local optimum. These strategies are usually seen in metaheuristics, such as Simulated Annealing (SA), Tabu Search (TS), Genetic Algorithm (GA) and so forth.

SA has been applied to a wide range of examination timetabling problems. Johnson (1990) used Simulated Annealing to solve the problems of University of the South Pacific (USP). A direct comparison with the previous manual approach was reported. It appears that the examination timetable planning has been simplified by using the proposed approach.

Bullnheimer (1998) formulated a small scale examination timetabling problem at the University of Magdeburg as a Quadratic Assignment Problem. SA with two different

neighborhood structures was then proposed. Some of the latest applications of SA in examination timetabling problems are presented by Thompson and Dowsland (1998), Merlot et al. (2003), Duong and Lam (2004) and Burke et al. (2004). The following table summarizes the applications of other metaheuristics in examination timetabling problems.

Table 2.5 Several applications of metaheuristics to the examination timetabling problem

No	Metaheuristics	Authors
1	Tabu Search	Boufflet and Negre (1996), White and Xie (2001) and Paquete and Stutzle (2002)
2	Genetic Algorithm	Burke et al. (1996), Ergül (1996) and Sheibani (2002)
3	Ant Colony Algorithm	Dowsland and Thompson (2005), Naji Azimi (2005)

2.4.2.4 Constraint Based Approaches

There are a number of papers from the Artificial Intelligence research community that use general systems for constraint representation. Here, examination timetabling problem is described as a set of examinations which require certain "resources" and there are several constraints need to be considered. There is no explicit minimization of maximization objective function. However, these approaches are usually integrated with other methods in order to reduce the time complexity. Examples of the applications of constraint based approaches in examination timetabling problems were presented by David (1998), Merlot et al. (2003) and Duang and Lam (2004).

2.5 Overview of Hybrid Algorithms

Timetabling problems belong to the class of combinatorial optimization problems (COP). A survey of related applications of combinatorial optimization is given by Grötschel (1991). In

general, a combinatorial optimization problem has a discrete finite search space *S* and a function *f*, that measures the quality of each solution in *S*. The main problem is to find $s^* = \arg\max_{s \in S} f(s)$ where *s* is a vector of decision variables and *f* is objective function that has to be maximized. The vector *s** is a global optimum. The neighborhood *N*(*s*) of a solution *s* in *S* is defined as the set of solutions which can be obtained from *s* by a *move*. Each solution is s denoted as s', where $s' \in N(s)$.

Two common techniques for solving timetabling problem are exact algorithms and heuristic methods. The former could guarantee finding an optimal solution. However, when problem size increases rapidly, the optimal solution could not be found within reasonable computation time. Perhaps, it could be due to a memory leak problem. On the other hand, heuristic methods are able to produce good enough solutions within reasonable computation time.

For many years, the main focus of research in timetabling problem was on the application of a single method to given problems. Recently, instead of comparing against some methods, some researchers have attempted to combine some methods. This recent area of research has become more important and viable due to increasing computational power. Some possible hybridizations will be described in the following sub-sections.

Talbi (2002) introduced the taxonomy of hybrid algorithms based on design and implementation issues. The hybrid algorithms are divided into low and high levels. In the low level hybridization, a given function of a metaheuristic is replaced by another metaheuristic. On the other hand, the different metaheuristics are self-contained in the high level hybrid algorithm. These two levels are further classified into relay and teamwork hybridization. In relay hybridization, a set of metaheuristics is applied one after another. Teamwork

hybridization uses many parallel cooperating agents, where each agent carries out a search in a solution space.

Other different classifications of hybrid algorithms were given by Blum and Roli (2003) and Puchinger and Raidl (2005). Different forms of hybridization proposed in Blum and Roli (2003) are (i) component exchange among metaheuristics, (ii) cooperative search and (iii) integrating metaheuristics and systematic methods. Puchinger and Raidl (2005) placed more emphasis on the combination of exact algorithms and metaheuristics, which are further categorized into two classes, namely collaborative and integrative combinations. This classification would be explained in Section 2.5.2.

2.5.1 Hybridization of Heuristics

Heuristic methods are classified into two categories: constructive and improvement heuristics. In Sections 2.4, we have described some applications of each category in timetabling problems.

One of the earlier papers that applied the idea of hybrid algorithms in the university course timetabling problem was presented by Weare et al. (1995). This paper proposed a hybrid Genetic Algorithm that combines a direct representation of the timetable and heuristic crossover operators. A hybrid of heuristic sequencing and evolutionary methods has also been presented by Burke and Newall (1999). This hybrid algorithm not only reduces the time taken to find the solution, but also improves the quality of the solutions.

Duong and Lam (2004) presented a hybrid method for the examination timetabling problem which consists of two phases: the first phase focuses on providing an initial solution by using a constraint programming approach, while the second phase focuses on determining crucial

cooling schedule parameters by applying SA. Merlot et al. (2003) proposed a new hybrid algorithm, consisting of three phases: a constraint programming phase for developing an initial solution, Simulated Annealing and hill climbing phases for improving the quality of solutions.

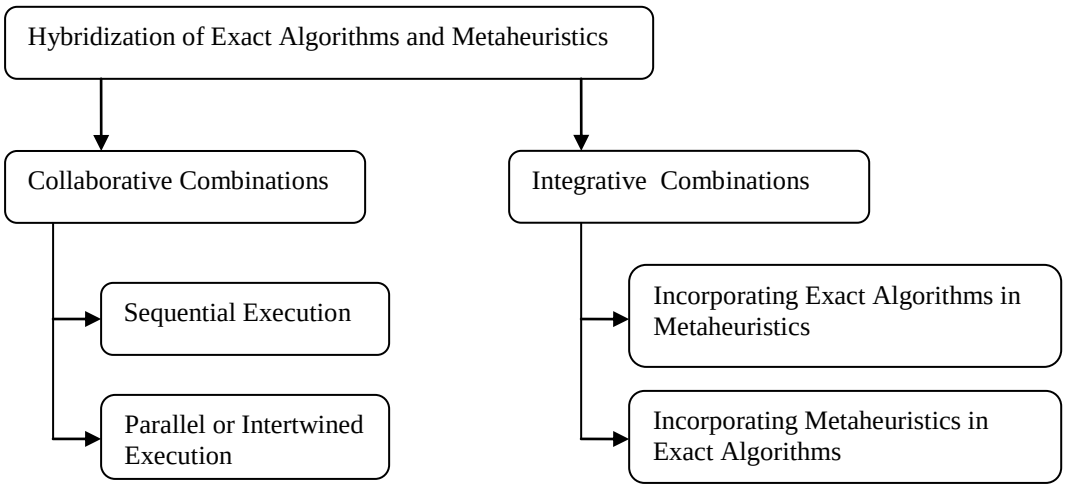
One of the latest applications of a hybrid algorithm in the timetabling problem is presented by Chiarandini et al. (2006). The proposed algorithm is mainly based on a framework consisting of the successive application of construction heuristics, variable neighborhood descent and Simulated Annealing. The paper also describes a hybrid metaheuristic algorithm involving a modified Simulated Annealing approach for solving the university course timetabling problem.

Another possible hybridization is to combine metaheuristic components with other components from other metaheuristics, known as hybrid metaheuristics. This recent area of research has become more important and viable due to increasing computational power. For instance, it may be desirable to have a memory element in the Simulated Annealing approach by incorporating the Tabu Search algorithm. Chiarandini and Stützle (2003) combined various construction heuristics, Tabu Search, variable neighborhood descent and Simulated Annealing for solving the course timetabling problem. Some examples of hybrid metaheuristics applied in other combinatorial optimization problems were proposed by Feng et al. (1993), Fox (1993), Bianchi et al. (2005) and Lim et al. (2005).

In the examination timetabling problem, sequential hybridization of Ant Colony System outperforms other types of hybridizations (Naji Azimi, 2005). Finally, one of the latest applications of hybrid metaheuristics in timetabling problems is presented by Rahoual and Saad (2006), which involving a combination of Genetic Algorithm and Tabu Search for tackling the timetabling problem.

2.5.2 Hybridization of Exact Algorithms and Heuristics/Metaheuristics

Puchinger and Raidl (2005) classify the hybridization of exact algorithms and metaheuristics into two different categories: collaborative and integrative combinations. Figure 2.4 gives an overview of this classification.



.

Figure 2.4 Major classification of exact/metaheuristic combinations

Collaborative algorithms refer to the algorithms that exchange information but are not part of each other or no algorithm is contained in another. Both exact and heuristic algorithms can be executed sequentially or in parallel. In sequential execution, either the exact algorithm is executed as a kind of pre-processing before the metaheuristic, or vice-versa. For instance, the original problem is relaxed and solved using CPLEX and the solution obtained is improved by a subsequent metaheurictic. Exact and metaheuristic algorithms can also be executed in a parallel or intertwined way. The basic idea is having a set of agents which all agents work in parallel on a set of shared memories. Each agent performs an optimization algorithm.

In integrative combinations, one technique is a subordinate embedded component of another technique. Two possibilities in the integrative combinations are:

- Incorporating Exact Algorithms in Metaheuristics

  Exact algorithms can be used to search neighborhoods in local search based metaheuristics. Such techniques are known as Very Large-Scale Neighborhood (VLSN) search (Ahuja et al., 2002).

- Incorporating Metaheuristics in Exact Algorithms

  In branch-and-price algorithm, the pricing of columns is sometimes done by means of heuristics including metaheuristics in order to speed up the whole optimization process. For instance, a heuristic column generation approach was applied to graph colouring problem by Filho and Lorena (2000).

The idea of hybrid algorithm has been widely applied in several combinatorial optimization problems such as flight network design problem (Büdenbender et al., 2000), production line scheduling problem (Clements et al., 1997), multiconstrained knapsack problem (Plateau et al., 2002), transport timetabling (Laplagne, 2005) and traveling salesman problem (Applegate et al., 1998 and Burke et al., 2001). It is highlighted that there are still many research opportunities to develop hybrid algorithms.

CHAPTER III

MATHEMATICAL PROGRAMMING MODELS FOR THE COURSE TIMETABLING PROBLEM

3.1 Introduction

Course timetabling problem can be classified into five sub-problems: teacher assignment, class-teacher timetabling, course scheduling, student scheduling and classroom assignment (Carter and Laporte, 1998). Although real life course timetabling problems always contain a combination of several sub-problems, we notice that many researchers only focus on one of the sub-problems. For instance, works of Andrew and Collins (1971), Schniederjans and Kim (1987) and Wang (2002) only focus on solving the teacher assignment problem, without considering the allocation of courses to time periods (the course scheduling problem).

On the other hand, the course scheduling problem only focuses on scheduling courses to time periods. It is often assumed that the allocation of teachers to courses has been performed before the actual scheduling of courses to time periods, as in the works of Al Yakoob et al. (2006, 2007), Daskalaki et al. (2004), Daskalaki and Birbas (2005) and Lewis and Paechter (2007).

In this research, we focus on a commonly encountered timetabling problem in many universities, and we call it the Teacher Assignment - Course Scheduling problem (TACS problem). The TACS problem is a combination of teacher assignment and course scheduling sub-problems, which considers requirements of both sub-problems simultaneously. The

primary problem faced is how to assign teachers to their preferred courses and then to schedule courses to time periods over a week based on the teachers' time preferences.

This chapter provides the basic and extended TACS mathematical programming models. The characteristics and the description of each proposed model are described in the subsequent sub-sections. The proposed models are then solved by commercial software, ILOG OPL Studio software. Finally, computational results based on some randomly generated data sets and conclusions are presented. The weaknesses of commercial software to solve large instances are also highlighted in the conclusion part.

3.2 The Basic TACS problem

3.2.1 Problem Description

In this sub-section, we start with a basic mathematical programming model that combines teacher assignment and course scheduling problems simultaneously at the university level. This timetabling problem that we address is peculiar to an engineering faculty of a university in Indonesia. A number of common definitions and terms for the basic problem are first explained as follows.

A course refers to a subject taught by a teacher within a week. Each course requires a time period in which each time period requires a certain number of consecutive hours. Before the new semester starts, teachers are requested to decide the courses they are willing to teach, along with their preferred time periods to teach the courses.

Although each university could have different requirements, the following description summarizes the most common and basic requirements that would be considered in our problem. The first requirement is considered as the soft constraint and it would be incorporated in the objective function that needs to be maximized, while the rest would be treated as hard constraints that cannot be violated. The requirements imposed in the basic problem are as follows:

1. Teacher preferences in terms of course and time period are maximized.
2. Each teacher has to teach at least one course and could not exceed the maximum number of courses allowed.
3. The number of courses taught could not exceed the number of classrooms available for each time period.
4. Each teacher can only teach one course, at most, at any time period.
5. All courses have to be scheduled.
6. Teachers will not be assigned courses that they are unable to teach.

3.2.2 The Mathematical Programming Model

We consider a set of teachers *I* who are willing to teach a set of courses *J*. All courses will be scheduled into the set of time periods *M*. The basic mathematical programming model is presented as an integer programming model with the following decision variable:

- X_{ijm} = 1 if teacher *i* teaches course *j* at time period *m*, 0 otherwise $(i \in I, j \in J, m \in M)$

A mathematical programming model for the basic TACS problem can be formulated as follows:

[TACS Model I]

Maximize $Z_{TACS\_I} = \sum_{i \in I} \sum_{j \in J} \sum_{m \in M} \left(PC_{ij} + PT_{im}^{I} \right) \times X_{ijm}$ (3.1)

subject to:

$1 \le \sum_{j \in J} \sum_{m \in M} X_{ijm} \le N_i \qquad (i \in I)$ (3.2)

$\sum_{i \in I} \sum_{j \in J} X_{ijm} \le C_m^{I} \qquad (m \in M)$ (3.3)

$\sum_{j \in J} X_{ijm} \le 1 \qquad (i \in I, m \in M)$ (3.4)

$\sum_{i \in I} \sum_{m \in M} X_{ijm} = 1 \qquad (j \in J)$ (3.5)

$X_{ijm} = 0 \qquad (i \in I, j \notin J_i, m \in M)$ (3.6)

$X_{ijm} \in \{0,1\} \qquad (i \in I, j \in J, m \in M)$ (3.7)

The objective function Z_{TACS_I} (equation 3.1) reflects a total preference function that needs to be maximized. It refers to the sum of value given by teacher *i* on the preference of being assigned to teach course *j* (PC_{ij}) and value given by teacher *i* on the preference of being assigned to teach at time period m (PT_{im}^{I}). Thus, the teacher can choose not only his course preference, but he can also indicate his time preference. In our model, we have assumed that these preferences are equally important. However, the values for these preferences can be scaled, or the preference function can be modified by adding different weights according to the actual timetabling scenarios encountered. In this basic model, it is assumed that the entire week is divided into 20 time periods.

Equation (3.2) ensures that each teacher i has to teach at least one course, while not allowing the teacher to teach more than the maximum number of courses allowed, N_i. Equation (3.3) represents the constraint on the number of classrooms available C_m^{I} during each time period m. Equation (3.4) ensures that each teacher *i* can only teach at most one course at any time period

m. Equation (3.5) states that each course *j* has to be allocated with a teacher at a particular time period *m*. Equation (3.6) ensures that teachers will not be assigned courses that they are unable to teach. Finally, constraint (3.7) imposes the 0-1 restrictions on the decision variables X_{ijm}.

3.2.3 Computational Results

The TACS Model I described above was solved by ILOG OPL Studio 3.7 on a 2.59GHz Pentium IV PC with 512MB RAM that runs in Microsoft Windows XP operating system. We generate randomly generated several data sets in such a way that the data sets correspond to differing values of four parameters.

Here, the four various parameters are the number of teachers $|I|$, the number of courses $|J|$, the maximum number of courses taught by each teacher $N_i (i \in I)$ and the number of classrooms available $C_m^{\mathrm{I}} (m \in M)$. The values of N_i and C_m^{I} may vary which depend on the characteristic of the institution. In our basic model, it is assumed that for $\forall i \in I$, $N_i = N$ and $\forall m \in M$, $C_m^{\mathrm{I}} = C$. The characteristics for each data set are summarized in Tables 3.1.

The following explanation describes how we define the preference values for each teacher. In terms of preference values, each teacher needs to indicate which courses and time periods are considered as the first priority, second priority and so on. This scenario can avoid a situation where teachers define very low values for all courses and time periods. These priorities would then be translated into numerical values, for instance, courses in the first priority would have a value of 100 and so on. For courses and time periods which are not selected, we simply set a zero value.

Table 3.1 Characteristics of data sets (TACS Model I)

Data set	Number of teachers $\|I\|$	Number of courses $\|J\|$	Maximum number of courses taught N	Number of classrooms available C
50_1	50	200	4	10
50_2	50	200	**5**	10
50_3	50	200	**6**	10
50_4	50	200	4	**15**
50_5	50	200	4	**20**
50_6	50	**250**	**5**	**13**
50_7	50	**300**	**6**	**15**
50_8	50	**350**	**7**	**18**
50_9	50	**400**	**8**	**20**
100_1	100	200	2	10
100_2	100	200	**3**	10
100_3	100	200	**4**	10
100_4	100	200	2	**15**
100_5	100	200	2	**20**
100_6	100	**250**	**3**	**13**
100_7	100	**300**	**3**	**15**
100_8	100	**350**	**4**	**18**
100_9	100	**400**	**4**	**20**

Each data set contains five randomly generated data instances. We set the number of time periods $|M|$ to be 20. This assumes that one week consists of 20 time periods in which each time period has a length of two and half hours (Figure 3.1).

		Day				
		1	2	3	4	5
Time	07.30 – 10.00	Period1	Period5	Period9	Period13	Period17
	10.00 - 12.30	Period2	Period6	Period10	Period14	Period18
	12.30 – 15.00	Period3	Period7	Period11	Period15	Period19
	15.00 - 17.30	Period4	Period8	Period12	Period16	Period20

Figure 3.1 Time periods in a week

In order to ensure feasibility of the problem instances being solved, the following formulae are used for calculating the maximum number of courses taught and number of classrooms available:

$$N \geq \left\lceil \frac{|J|}{|I|} \right\rceil \tag{3.8}$$

$$C \geq \left\lceil \frac{|J|}{|M|} \right\rceil \tag{3.9}$$

Table 3.2 shows the results obtained when the number of teachers is set to 50. It summarizes the average optimal objective function values obtained and the average CPU time required to obtain the solutions.

Data set 50_1 is used as the 50-teacher base data set by setting equations (3.8) and (3.9) to equality. For data sets 50_2 to 50_5, we change the value of either the maximum number of courses taught N or the number of classrooms available C in the 50-teacher base data set. In general, we observe that some of the average CPU time required decreases when the maximum number of courses taught N or the number of classrooms available C increases.

The possible reason for this situation is when we increase the maximum number of courses taught N or the number of classrooms available C, it becomes easier for equations (3.2) and (3.3) to be satisfied, which may result in decreased CPU time when finding the optimal solution for these problem instances. When the maximum number of courses taught or the number of classrooms available is increased, the average objective value of the optimal solution is increased, meaning more courses and time periods are assigned to each teacher.

Table 3.2 Computational results by OPL Solver when number of teachers is 50

Data set	Objective function value	CPU time (seconds)	Average objective function value	Average CPU time (seconds)
50_1(1)	34250	19.06	34167.40	19.30
50_1(2)	34097	23.17		
50_1(3)	34097	18.40		
50_1(4)	34424	18.26		
50_1(5)	33969	17.63		
50_2(1)	35770	15.18	35645.80	16.77
50_2(2)	35257	17.44		
50_2(3)	35707	17.57		
50_2(4)	35707	17.03		
50_2(5)	35788	16.63		
50_3(1)	35759	16.55	35847.60	16.93
50_3(2)	35510	18.18		
50_3(3)	35845	17.16		
50_3(4)	36097	15.94		
50_3(5)	36027	16.84		
50_4(1)	34572	17.53	34797.40	15.77
50_4(2)	34727	15.50		
50_4(3)	34410	15.32		
50_4(4)	35290	14.52		
50_4(5)	34988	15.99		
50_5(1)	34876	15.44	34968.40	15.40
50_5(2)	34930	15.72		
50_5(3)	35621	15.47		
50_5(4)	34979	15.41		
50_5(5)	34436	14.97		
50_6(1)	40889	52.55	41185.20	29.92
50_6(2)	40741	24.08		
50_6(3)	41011	24.73		
50_6(4)	41913	24.06		
50_6(5)	41372	24.20		
50_7(1)	47681	94.32	48083.00	54.81
50_7(2)	47675	42.69		
50_7(3)	48674	48.58		
50_7(4)	48685	42.49		
50_7(5)	47700	45.94		
50_8(1)	53300	177.69	54139.80	119.48
50_8(2)	55039	115.59		
50_8(3)	53777	98.64		
50_8(4)	54770	115.97		
50_8(5)	53813	89.52		
50_9(1)	60176	158.98	59709.80	204.99
50_9(2)	60159	203.38		
50_9(3)	60159	195.85		
50_9(4)	59329	288.12		
50_9(5)	58726	178.62		

The following figure illustrates part of the optimal solution obtained for data set 50_1(1) including a numerical example. For instance, Teacher1 is allocated to teach Course 171 in Period 2, Course 177 in Period 5 and so on.

Period	1	2	…	5	6	7	8	9	…	20
Teacher1		171		177		166				
Teacher2					92					
Teacher3	47									
:										
:										
Teacher50		170						16		

Figure 3.2 Part of the result of data set 50_1(1)

Data sets 50_6 to 50_9 consider the case when the number of courses offered is increased from 250 to 400 courses. We notice that when the number of courses is increased, the average CPU time required increases rapidly (see Figure 3.3). In general, we conclude that the average CPU time will increase when the size of the problem increases.

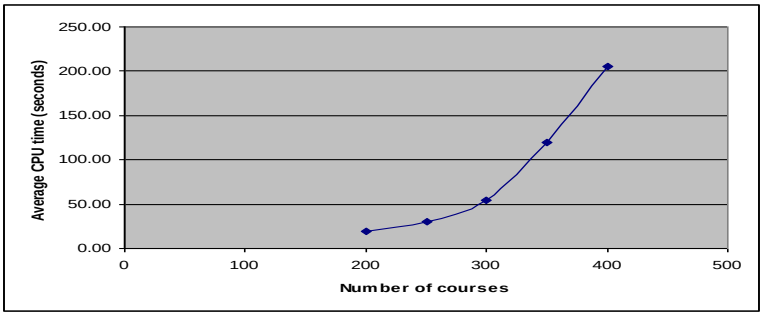


Figure 3.3 Plot of average CPU time for data sets 50_1, 50_6 to 50_9

Table 3.3 Computational results by OPL Solver when number of teachers is 100

Data set	Objective function value	CPU time (seconds)	Average objective function value	Average CPU time (seconds)
100_1(1)	36943	194.93	36887.80	190.23
100_1(2)	36916	170.86		
100_1(3)	36906	142.38		
100_1(4)	36653	152.68		
100_1(5)	37021	290.29		
100_2(1)	37912	185.00	37938.20	208.10
100_2(2)	38085	190.08		
100_2(3)	38085	203.60		
100_2(4)	37634	237.87		
100_2(5)	37975	223.96		
100_3(1)	38197	206.56	38342.40	216.50
100_3(2)	38468	251.53		
100_3(3)	38468	146.63		
100_3(4)	38494	217.62		
100_3(5)	38085	260.15		
100_4(1)	37325	219.16	37371.00	176.10
100_4(2)	37251	99.05		
100_4(3)	37337	183.03		
100_4(4)	37649	145.49		
100_4(5)	37293	233.79		
100_5(1)	37007	76.17	37290.00	139.18
100_5(2)	37007	162.95		
100_5(3)	37551	161.09		
100_5(4)	37647	73.85		
100_5(5)	37238	221.86		
100_6(1)	45744	342.61	45667.60	369.13
100_6(2)	45717	332.26		
100_6(3)	44973	303.11		
100_6(4)	45952	491.75		
100_6(5)	45952	375.91		
100_7(1)	51819	488.21	52015.40	547.42
100_7(2)	52573	545.95		
100_7(3)	52573	818.63		
100_7(4)	51937	322.21		
100_7(5)	51175	562.08		
100_8(1)	60611	1336.72	60360.20	1355.55
100_8(2)	60479	1728.42		
100_8(3)	60479	1424.24		
100_8(4)	59773	1075.08		
100_8(5)	60459	1213.29		
100_9(1)	64963	1670.30	65259.80	1818.39
100_9(2)	65753	1815.83		
100_9(3)	65938	1956.49		
100_9(4)	65734	1441.68		
100_9(5)	63911	2207.66		

The next set of results looks into the effect of increasing the number of teachers to 100. We have to adjust the maximum number of courses taught and the number of classrooms available accordingly. Table 3.3 shows the computational results when the number of teachers is set to 100 with data set 100_1 as the 100-teacher base data set. The configurations of the other data sets 100_2 to 100_9 are generated in a similar way as that of those data sets 50_2 to 50_9.

From Table 3.3, we observe a decrease in the average CPU time required when the number of classrooms available increases. When the number of courses is increased as in data sets 100_6 to 100_9, we observe that the average CPU time required increases rapidly (see Figure 3.4 for illustration). In addition, we observe that the average CPU time needed for solving the 100-teacher problem instances is much larger than that for the 50-teacher problem instances when we compare the results in Tables 3.2 and 3.3.

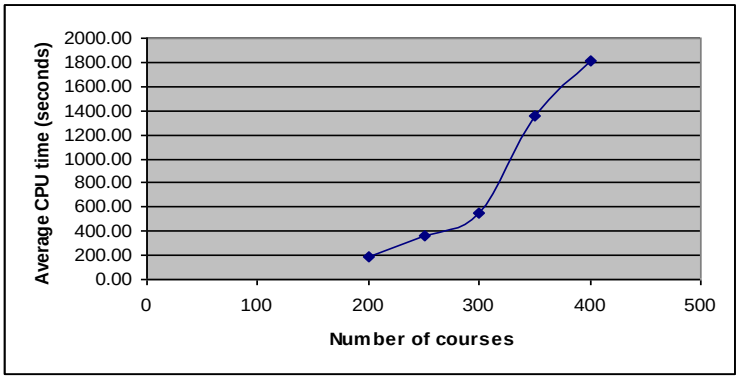


Figure 3.4 Plot of average CPU time for data sets 100_1, 100_6 to 100_9

Another observation of interest is the distribution of the number of courses taught by each teacher with respect to the maximum number of courses taught that we specify (see Table 3.4).

Table 3.4 Distribution of average number of courses taught by each teacher

Data set	Maximum number of courses taught	Average percentage of teachers teaching the following number of courses (%)						Variance
		1	2	3	4	5	6	
50_1	4	0.0	0.0	0.0	100.0	0.0	0.0	0.00
50_2	5	0.0	2.4	20.4	52.0	25.2	0.0	0.55
50_3	6	0.0	1.2	26.0	47.6	22.0	3.2	0.66
100_1	2	0.0	100.0	0.0	0.0	0.0	0.0	0.00
100_2	3	18.4	62.8	18.8	0.0	0.0	0.0	0.37
100_3	4	20.0	60.6	18.6	0.8	0.0	0.0	0.42

The variances obtained are reasonably small, with only a small percentage of teachers having a very small or a very large number of courses taught. Thus, the optimal solutions obtained appear to have the property of sharing the similar number of courses taught among teachers. For example, in data sets 50_1, 50_2, 50_3, the number of teachers and courses are 50 and 200, respectively. Most of teachers will tech 4 courses.

In this section, the basic TACS problem has been presented in detail. In the following sections, we extend the basic model into more complicated models by incorporating several additional requirements or constraints.

3.3 The First Extended TACS problem

3.3.1 Problem Description

As explained earlier, the basic model only considers the following situations: each course can only be taught by one teacher and conducted only once a week. Here, the first extended model, namely TACS Model II, would accommodate the following different requirements:

- Assuming the number of students registered to a particular course is probably greater than the capacity of a classroom. It would be necessary to divide a particular course into several

course sections that are taught by the same or different teachers. In order to decide which courses have to be divided to course sections, the university needs to forecast the number of students who are going to take a particular course by referring to the university data base.

- A course can be taught more than once within a week. However, at any particular time period, only one course section can be taught.
- Since a course can be divided into several sections, each course can be taught by more than one teacher. The minimum and maximum number of teachers who can teach for each course depend on the number of sections offered for each course.. We only restrict each course section can only be taught by one teacher.

3.3.2 The Mathematical Programming Model

TACS Model II is presented as an integer programming model with the following additional decision variables:

- P_{ij} = 1 if teacher *i* teaches course *j*, 0 otherwise $\left(i \in I, j \in J\right)$
- X_{ijkm} = 1 if teacher *i* teaches course *j* section *k* at time period *m*, 0 otherwise $\left(i \in I, j \in J, k \in K_j, m \in M\right)$

A mathematical programming model for the timetabling problem that we address can then be formulated as follows:

[TACS Model II]

$$\text{Maximize } Z_{TACS\_II} = \sum_{i \in I} \sum_{j \in J} PC_{ij} \times P_{ij} + \sum_{i \in I} \sum_{j \in J} \sum_{k \in K_j} \sum_{m \in M} PT_{im}^{\mathrm{I}} \times X_{ijkm} \tag{3.10}$$

subject to:

$$\sum_{i \in I} \sum_{k \in K_j} X_{ijkm} \le 1 \qquad \left(j \in J, m \in M\right) \tag{3.11}$$

$$P_{ij} = \left\lceil \sum_{k \in K_j} \frac{\sum_{m \in M} X_{ijkm}}{Sec_j} \right\rceil \qquad (i \in I, j \in J) \qquad (3.12)$$

$$LT_j \le \sum_{i \in I} P_{ij} \le UT_j \qquad (j \in J) \qquad (3.13)$$

$$1 \le \sum_{j \in J} P_{ij} \le N_i \qquad (i \in I) \qquad (3.14)$$

$$\sum_{j \in J} \sum_{k \in K_j} X_{ijkm} \le 1 \qquad (i \in I, m \in M) \qquad (3.15)$$

$$\sum_{i \in I} \sum_{j \in J} \sum_{k \in K_j} X_{ijkm} \le C_m^{\mathrm{I}} \qquad (m \in M) \qquad (3.16)$$

$$\sum_{i \in I} \sum_{k \in K_j} \sum_{m \in M} X_{ijkm} = Sec_j \qquad (j \in J) \qquad (3.17)$$

$$\sum_{i \in I} \sum_{m \in M} X_{ijkm} = 1 \qquad (j \in J, k \in K_j) \qquad (3.18)$$

$$X_{ijkm} = 0 \qquad (i \in I, j \notin J_i, k \in K_j, m \in M) \qquad (3.19)$$

$$X_{ijkm} \in \{0,1\} \qquad (i \in I, j \in J, k \in K_j, m \in M) \qquad (3.20)$$

The objective is to maximize the course and time preference of all teachers. It is reflected in equation (3.10). Equation (3.11) ensures that for a particular course, only one section can be conducted in every period. Equation (3.12) represents the relationship between variables P_{ij} and X_{ijkm}. It indicates that if teacher *i* teach at least one section of course *j*, the value of P_{ij} would be 1, meaning that teacher *i* teaches course *j*. Equation (3.13) limits the number of teachers who can teach for each course. Equation (3.14) ensures that each teacher has to teach between the minimum and maximum number of courses taught.

Equation (3.15) ensures that each teacher can only teach at most one course section at a particular time period. Equation (3.16) represents the constraint on the number of classrooms available during each time period. Equation (3.17) states that all sections for a particular course must be scheduled. Equation (3.18) assumes that each course section can only be taught

by one teacher. Equation (3.19) ensures that teachers will not be assigned courses that he/she is unable to teach. Finally, constraint (3.20) imposes the 0-1 restrictions on the decision variables X_{ijkm}.

Note that equation (3.12) involves nonlinear functions of the decision variables but these can always be linearized by adding the following constraint:

$$Sec_j \times (\varepsilon + P_{ij} - 1) \le \sum_{k \in K_j} \sum_{m \in M} X_{ijkm} \le P_{ij} \times Sec_j \qquad (i \in I, j \in J) \tag{3.21}$$

Here, ε is any positive number such that $\varepsilon < \min_j \{1/Sec_j\}$.

Preposition 3.1 If $\varepsilon < \min_j \{1/Sec_j\}$, then equation (3.21) holds.

Proof.

From equation (3.21), one obtains $\varepsilon \le \dfrac{\sum_{k \in K_j} \sum_{m \in M} X_{ijkm}}{Sec_j} + 1 - P_{ij}$ for ($i \in I$, $j \in J$)

Case 1. Suppose teacher *i* teaches course *j*, thus $\sum_{k \in K_j} \sum_{m \in M} X_{ijkm} \ge 1$ as well as

$$\varepsilon < \frac{1}{Sec_j} \le \frac{\sum_{k \in K_j} \sum_{m \in M} X_{ijkm}}{Sec_j} \le \frac{\sum_{k \in K_j} \sum_{m \in M} X_{ijkm}}{Sec_j} + 1 - P_{ij}$$

Case 2. Suppose teacher *i* does not teach course *j*, thus $P_{ij} = \sum_{k \in K_j} \sum_{m \in M} X_{ijkm} = 0$ as well as

$$\varepsilon < 1 = \frac{\sum_{k \in K_j} \sum_{m \in M} X_{ijkm}}{S_j} + 1 - P_{ij}$$

By referring to equation (3.21), $\left(\varepsilon + P_{ij} - 1\right) \leq \frac{\sum_{k \in K_j} \sum_{m \in M} X_{ijkm}}{Sec_j} \leq P_{ij}$, the value of $P_{ij} = \left\lceil \frac{\sum_{k \in K_j} \sum_{m \in M} X_{ijkm}}{Sec_j} \right\rceil, \left\lceil \frac{\sum_{k \in K_j} \sum_{m \in M} X_{ijkm}}{Sec_j} \right\rceil + 1, \left\lceil \frac{\sum_{k \in K_j} \sum_{m \in M} X_{ijkm}}{Sec_j} \right\rceil + 2, \ldots$. By contradiction, if $P_{ij} \geq \left\lceil \frac{\sum_{k \in K_j} \sum_{m \in M} X_{ijkm}}{Sec_j} \right\rceil + 1$, then $\frac{\sum_{k \in K_j} \sum_{m \in M} X_{ijkm}}{Sec_j} \geq \varepsilon + P_{ij} - 1 \geq \varepsilon + \left\lceil \frac{\sum_{k \in K_j} \sum_{m \in M} X_{ijkm}}{Sec_j} \right\rceil > \left\lceil \frac{\sum_{k \in K_j} \sum_{m \in M} X_{ijkm}}{Sec_j} \right\rceil$

Finally, we conclude that $\varepsilon < \min_j \left\{ 1 / Sec_j \right\}$.

3.3.3 Computational Results

TACS Model II was solved by ILOG OPL Studio 3.7 on the earlier processor and operating system used for solving TACS Model I. Several random data sets were generated in such a way that they correspond to differing values of the following parameters: the number of teachers $|I|$, the number of courses $|J|$, the maximum number of courses taught N and the number of classrooms available C.

Two additional parameters, the number of sections Sec_j and the number of time periods $|M|$, are set to constant values. Each data set contains five different randomly generated data instances. The following table shows the detail characteristics of data sets. The number of sections for each course and the number of time period per week are set to 3 and 20, respectively.

Table 3.5 Characteristics of data sets (TACS Model II)

Data set	Number of teachers $\|I\|$	Number of courses $\|J\|$	Maximum number of courses taught N	Number of classrooms available C
25_1	25	75	3	7
25_2	25	75	4	7
25_3	25	75	5	7
50_1	50	150	3	13
50_2	50	150	4	13
50_3	50	150	5	13
75_1	75	225	3	20
75_2	75	225	4	20
75_3	75	225	5	20

Table 3.6 Computational results by OPL Solver when number of teachers is 25

Data set	Objective function value	CPU time (seconds)	Average objective function value	Average CPU time (seconds)
25_1(1)	12080	3.73	12474	4.20
25_1(2)	12170	4.88		
25_1(3)	13060	3.73		
25_1(4)	12670	3.87		
25_1(5)	12390	4.80		
25_2(1)	13750	16.65	14238	6.91
25_2(2)	14000	3.61		
25_2(3)	14930	4.69		
25_2(4)	14360	4.18		
25_2(5)	14150	5.41		
25_3(1)	14440	3.37	14874	4.28
25_3(2)	14640	5.36		
25_3(3)	15770	4.20		
25_3(4)	14810	3.93		
25_3(5)	14710	4.56		

Table 3.6 shows the computational results when the number of teachers is set to 25. It summarizes the average optimal objective function values obtained and the average CPU time required to obtain the solutions. In general, we observe that when the maximum number of courses taught is increased, the objective function value for each instance and the average

objective value of the optimal solution are increased. More courses and time periods would be assigned to each teacher.

Table 3.7 Computational results by OPL Solver when number of teachers is 50

Data set	Objective function value	CPU time (seconds)	Average objective function value	Average CPU time (seconds)
50_1(1)	24730	80.10	24572	48.06
50_1(2)	24540	19.61		
50_1(3)	23740	46.65		
50_1(4)	24830	52.96		
50_1(5)	25020	40.95		
50_2(1)	28130	38.30	28018	75.47
50_2(2)	27950	53.51		
50_2(3)	27230	64.77		
50_2(4)	28060	85.15		
50_2(5)	28720	135.60		
50_3(1)	28990	19.93	28964	33.71
50_3(2)	28760	29.51		
50_3(3)	28330	25.42		
50_3(4)	28910	42.94		
50_3(5)	29830	50.74		

Table 3.8 Computational results by OPL Solver when number of teachers is 75

Data set	Objective function value	CPU time (seconds)	Average objective function value	Average CPU time (seconds)
75_1(1)	38620	1409.13	37442	1711.49
75_1(2)	37180	1264.74		
75_1(3)	37010	1720.73		
75_1(4)	36750	2142.15		
75_1(5)	37650	2020.72		
75_2(1)	43950	1635.38	42562	1786.72
75_2(2)	42240	1749.48		
75_2(3)	42070	1990.90		
75_2(4)	41710	1567.05		
75_2(5)	42840	1990.80		
75_3(1)	45510	1077.42	43988	1514.37
75_3(2)	43680	1667.73		
75_3(3)	43230	1646.54		
75_3(4)	43100	1633.72		
75_3(5)	44420	1546.42		

Similar observations have been found for other data sets with the number of teachers are set to 50 and 75 (Tables 3.7 and 3.8). The average CPU time needed for solving large instances is much larger than that for the 25-teacher problem instances. From Table 3.8, we observe that we require almost 30 minutes for solving the 75-teacher problem instances.

Table 3.9 Distribution of average number of courses taught by each teacher

Data Set	Average percentage of teachers teaching following number of courses (%)				
	1	2	3	4	5
25_1	0.0	0.0	100.0	-	-
25_2	0.0	0.0	0.0	100.0	-
25_3	0.0	1.6	5.6	21.6	71.2
50_1	0.0	0.0	100.0	-	-
50_2	0.0	0.0	0.8	99.2	-
50_3	0.0	1.2	11.6	33.2	54.0
75_1	0.0	0.0	100.0	-	-
75_2	0.0	0.0	0.5	99.5	-
75_3	0.0	0.8	13.1	31.7	54.4

For further analysis, we observe the distribution of average number of courses taught by each teacher with respect to the maximum value specified (see Table 3.9). If we increase the maximum number of courses taught, there is a tendency that most of the teachers will teach as many as the maximum number of courses taught N_i.

For example, when the maximum number of courses taught is set to 3 courses (data set 25_1), all teachers will teach 3 courses. Similar observation when we increase the maximum number of courses taught to 4 courses (data set 25_2), each teacher will teach 4 different courses. It could be due to the following requirements: the numbers of teachers teach a particular course can be more than one teacher and the minimum number of teachers required for a particular course. For instance, courses with 4 sections require at least two teachers to teach. It turns out that more teachers will be assigned to courses. At the end, more courses would be assigned to

each teacher as long as the number of courses taught does not exceed the maximum number of courses taught.

Table 3.10 Number of teachers allocated to a particular course

Data set	Number of teachers $\|I\|$	Number of courses $\|J\|$	Maximum number of courses taught N	The percentage of courses taught by	
				1 teacher	2 teachers
25_1	25	75	3	100.0	0.0
25_2	25	75	4	66.7	33.3
25_3	25	75	5	45.9	54.1
50_1	50	150	3	100.0	0.0
50_2	50	150	4	66.9	33.1
50_3	50	150	5	53.3	46.7
75_1	75	225	3	100.0	0.0
75_2	75	225	4	66.8	33.2
75_3	75	225	5	53.4	46.6

From Table 3.10, if the number of courses $|J|$ divided by the number of teachers $|I|$ is equal to the maximum number of courses taught per teacher N, each course would be taught by one teacher. On the other hand, if we increase the maximum number of courses taught value, we notice that more than one teacher would be able to teach a particular course. The more we increase the value, the more number of teachers would teach a particular course.

3.4 The Second Extended TACS problem

3.4.1 Problem Description

The second extended model, namely TACS Model III, includes several additional requirements which are comparable to those occurring in an engineering faculty of a university in Indonesia. Similar to TACS Model II, due to the capacity of the classrooms and the number

of students registered, some courses have to be taught repeatedly by the same teacher or by different teachers. Each of these repeated courses is known as a course section.

In previous models (TACS Model I and II), a week is assumed to be divided into several time periods with the same number of hours (Figure 3.1). In TACS Model III, time periods are considered on a day-hour basis and courses can be conducted at any time period in order to increase teaching flexibility (see Figure 3.5 for illustration). Each course/course section requires a certain number of consecutive time periods per week.

Similar to both TACS Model I and TACS Model II, teachers are requested to decide the courses they are willing to teach before the new semester starts, along with their preferred days and time periods to teach the courses. We adopt a similar approach by formulating TACS Model III that considers both teacher assignment and course scheduling simultaneously.

The primary problem faced is to assign teachers to their preferred courses and course sections and then to schedule course sections to time periods over a week based on the teachers' preferences (Figure 3.5).

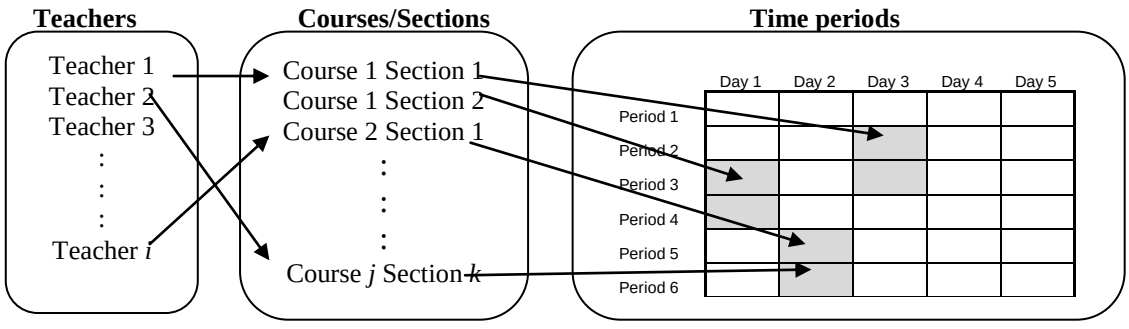


Figure 3.5 TACS Model III

The university timetabling problem has special features that highly depend on the university's characteristics, such as the courses taught, the teachers and the availability of resources as well.

Although each university might have different requirements, the following description summarizes the most common requirements that will be regarded as hard constraints that cannot be violated.

There are some requirements which are similar to those of TACS Model II (Section 3.3.1). Here, the following additional requirements pertaining to TACS Model III are listed:

- In order to avoid high workload for each teacher or unbalanced teaching load, all the course sections taught by a teacher have to be spread evenly throughout a week.
- Each course section requires a certain number of time periods to be scheduled consecutively.
- Another specific requirement in the context of an engineering faculty in a university in Indonesia is the number of sections for a particular course that can be conducted each day. In particular, this is restricted to only one section that can be conducted each day so that all sections can be spread evenly throughout the week. The main reason is to increase the chance for students to select the courses. For instance, Courses *A* (Section 1) and *B* (Section 1) are taught on Monday (time period 1). Students definitely cannot take both courses at the same time. However, another section for each course (Section 2) would be scheduled on a different day so students may consider to take another course (either Course A or B) on a different day. We assume that the number of sections for a particular course is less than or equal to the number of days in a week.

3.4.2 The Mathematical Programming Model

The timetable is in the form of a weekly schedule. A week is further partitioned into a set of days (*L*) and time periods (*M*). In this study, each time period *m* is assumed to be of the same duration. Each section *k* of course *j* $\left(k \in K_j\right)$ has to be scheduled into time periods based on the

number of time periods required, H_j. Each time period $m \in M$ on day $l \in L$ has a maximum number of classrooms available, C_{lm}^{III}. For simplicity, we again assume that C_{lm}^{III} is a constant, i.e., $C_{lm}^{\text{III}} = C$ for all l and all m, where C is a positive integer.

The decision variables needed in the model are defined next:

X_{ijklm} = 1 if teacher i teaches course j section k on day l and at time period m; 0 otherwise $\left(i \in I, j \in J, k \in K_j, l \in L, m \in M\right)$

Y_{ijkl} = 1 if teacher i teaches course j section k on day l; 0 otherwise $\left(i \in I, j \in J, k \in K_j, l \in L\right)$

U_{ijklm} = 1 if teacher i teaches course j section k on day l and starts at time period m; 0 otherwise $\left(i \in I, j \in J, k \in K_j, l \in L, m \in M\right)$

P_{ij} = 1 if teacher i teaches course j; 0 otherwise $\left(i \in I, j \in J\right)$

L_i = number of course sections taught by teacher i $\left(i \in I\right)$

V_i = number of course sections taught by teacher i per day $\left(i \in I\right)$

A mathematical programming model for the timetabling problem can then be formulated as follows:

[TACS Model III]

$$\text{Maximize } Z_{TACS\_\text{III}} = \sum_{i \in I} \sum_{j \in J} PC_{ij} \times P_{ij} + \sum_{i \in I} \sum_{j \in J} \sum_{k \in K_j} \sum_{l \in L} \sum_{m \in M} PT_{ilm}^{\text{III}} \times X_{ijklm} \tag{3.22}$$

subject to:

$$\sum_{i \in I} \sum_{k \in K_j} X_{ijklm} \le 1 \qquad \left(j \in J, l \in L, m \in M\right) \tag{3.23}$$

$$P_{ij} = \left\lceil \sum_{k \in K_j} \sum_{l \in L} \frac{Y_{ijkl}}{Sec_j} \right\rceil \qquad \left(i \in I, j \in J\right) \tag{3.24}$$

$$1 \le \sum_{j \in J} P_{ij} \le N_i \qquad \left(i \in I\right) \tag{3.25}$$

$$LT_j \le \sum_{i \in I} P_{ij} \le UT_j \qquad (j \in J) \tag{3.26}$$

$$\sum_{m \in M} X_{ijklm} = Y_{ijkl} \times H_j \qquad (i \in I, j \in J, k \in K_j, l \in L) \tag{3.27}$$

$$\sum_{i \in I} \sum_{k \in K_j} Y_{ijkl} \le 1 \qquad (j \in J, l \in L) \tag{3.28}$$

$$\sum_{j \in J} \sum_{k \in K_j} X_{ijklm} \le 1 \qquad (i \in I, l \in L, m \in M) \tag{3.29}$$

$$\sum_{i \in I} \sum_{j \in J} \sum_{k \in K} X_{ijklm} \le C_{lm}^{\mathrm{III}} \qquad (l \in L, m \in M) \tag{3.30}$$

$$\sum_{i \in I} \sum_{k \in K_j} \sum_{l \in L} Y_{ijkl} = Sec_j \qquad (j \in J) \tag{3.31}$$

$$\sum_{i \in I} \sum_{l \in L} Y_{ijkl} = 1 \qquad (j \in J, k \in K_j) \tag{3.32}$$

$$X_{ijklm} = 0 \qquad (i \in I, j \notin J_i, k \in K_j, l \in L) \tag{3.33}$$

$$\sum_{j \in J} \sum_{k \in K_j} \sum_{l \in L} Y_{ijkl} = L_i \qquad (i \in I) \tag{3.34}$$

$$V_i = \left\lceil \frac{L_i}{|L|} \right\rceil \qquad (i \in I) \tag{3.35}$$

$$\sum_{j \in J} \sum_{k \in K_j} Y_{ijkl} \le V_i \qquad (i \in I, l \in L) \tag{3.36}$$

$$\sum_{t=0}^{(H_j - 1)} X_{ijkl(m+t)} \ge H_j \times U_{ijklm} \qquad (i \in I, j \in J, k \in K_j, l \in L, m \in \{1, \ldots, |M| - H_j + 1\}) \tag{3.37}$$

$$\sum_{i \in I} \sum_{l \in L} \sum_{m=1}^{(|M| - H_j + 1)} U_{ijklm} = 1 \qquad (j \in J, k \in K_j) \tag{3.38}$$

$$\sum_{i \in I} \sum_{l \in L} \sum_{m \in M} X_{ijklm} = H_j \qquad (j \in J, k \in K_j) \tag{3.39}$$

$$U_{ijklm} = 0 \qquad (i \in I, j \in J, k \in K_j, l \in L, m \in \{|M| - H_j + 2, \ldots, |M|\}) \tag{3.40}$$

$$U_{ijklm} = 0 \qquad (i \in I, j \notin J_i, k \in K_j, l \in L, m \in \{1, \ldots, |M| - H_j + 1\}) \tag{3.41}$$

$$X_{ijklm} \in \{0,1\} \qquad (i \in I, j \in J, k \in K_j, l \in L, m \in M) \tag{3.42}$$

$$Y_{ijkl} \in \{0,1\} \qquad (i \in I, j \in J, k \in K_j, l \in L) \tag{3.43}$$

$$U_{ijklm} \in \{0,1\} \qquad (i \in I, j \in J, k \in K_j, l \in L, m \in M) \tag{3.44}$$

$$P_{ij} \in \{0,1\} \qquad (i \in I, j \in J) \tag{3.45}$$

$$L_i \in Z^+ \qquad (i \in I) \tag{3.46}$$

$$V_i \in Z^+ \qquad (i \in I) \tag{3.47}$$

The objective function (3.22) is similar to that of equation (3.1). It refers to the sum of value given by teacher *i* on the preference of being assigned to teach course *j* (PC_{ij}) and value given by teacher *i* on the preference of being assigned to teach on day *l* time period *m* (PT_{ilm}^{III}).

Equation (3.23) ensures that at most one section can be taught in every time period for a particular course *j*. Equation (3.24) ensures that the variable P_{ij} takes the value of 1 when teacher *i* teaches at least one section of course *j*; otherwise, it would be 0. Equation (3.25) ensures that each teacher has to teach at least one course. Equation (3.25) also represents the constraint that each teacher cannot teach more than the maximum number of courses taught. Teachers would not teach more than the maximum number of courses allowed although he shows great interests on many courses.

Equation (3.26) restricts for each course the number of teachers who could teach it. Equations (3.25) and (3.26) try to reduce a situation where more courses would be assigned to teachers who show great interests on many courses compared to those teachers who do not show interest in most of the courses. For example, course *A* with 4 course sections, the minimum number of teachers teach this particular course is set to two. In this case, although only one teacher shows a great interest on that course, the university has to find another teacher with less interest in order to satisfy the minimum number of teacher required to teach that particular course.

Equation (3.27) shows the relationship between variables Y_{ijkl} and X_{ijklm}. If teacher *i* teaches course *j* section *k* during H_j time periods on day *l*, the value of Y_{ijkl} is equal to 1. Equation (3.28) ensures that for any course *j*, at most one section can be conducted each day. Equation (3.29) ensures that each teacher can only be assigned at most one course section at any time period.

Equation (3.30) prevents the total number of course sections conducted per time period from exceeding the number of classrooms available, C_{lm}^{III}. Equation (3.31) states that all course sections for each course must be scheduled in the timetable. Equation (3.32) ensures that each course section can only be taught by one teacher, while equation (3.33) ensures that teachers will not be assigned courses that they are unable to teach.

Equation (3.34) calculates the number of course sections taught by each teacher and equation (3.35) determines the number of course sections taught per day for each teacher, rounded upwards to the nearest integer. Equation (3.36) helps to spread evenly all the course sections taught by each teacher throughout a week.

Equations (3.37), (3.38) and (3.39) deal with the consecutive requirements. With these constraints, if course *j* section *k* taught by teacher *i* requires H_j consecutive time periods and the teacher is assigned to a given period of day *l* as the first period to be taught for the course section, then the teacher will be assigned to the following (H_j - 1) periods. In order to facilitate the modeling of this requirement, the variable U_{ijklm} is introduced.

Equation (3.37) expresses the logic that each course section has to be scheduled and taught by a teacher in H_j time periods consecutively. If the start of section *k* of course *j* taught by teacher *i* is assigned to time period m_1 of day *l*, i.e., the variable U_{ijklm_1} takes the value of 1, then the

following ($H_j - 1$) time periods have to be assigned to the same course section. Equation (3.38) ensures that there is only one starting time period for each course section, and equation (3.39) further ensures that the number of time periods allocated to each course section meets its requirement.

For variables U_{ijklm}, additional constraints (3.40) and (3.41) are introduced to ensure that a course section could not be started in certain time periods if the remaining time periods are less than the number of time periods required and teachers will not be assigned certain time periods for courses that they are unable to teach, respectively. Finally, constraints (4.42), (4.43), (4.44) and (4.45) impose the 0-1 restrictions for the decision variables X_{ijklm}, Y_{ijkl}, U_{ijklm} and P_{ij}, respectively; while constraints (4.46) and (4.47) represent the nonnegative and integrality requirements for the L_i and V_i variables.

Even though equations (3.24) and (3.35) involve nonlinear functions of the decision variables, they can be rewritten as a linear model TACS′ Model III by introducing additional constraints (3.48) and (3.49) as follows:

$$Sec_j \times (\varepsilon + P_{ij} - 1) \le \sum_{k \in K_j} \sum_{l \in L} Y_{ijkl} \le P_{ij} \times Sec_j \qquad (i \in I, j \in J) \tag{3.48}$$

$$|L| \times (\varepsilon + V_i - 1) \le L_i \le V_i \times |L| \qquad (i \in I) \tag{3.49}$$

Here, ε is any positive number such that $\varepsilon < \min\left\{\min_j \{1/Sec_j\}, 1/|L|\right\}$.

Preposition 3.2. If $\varepsilon < \min\left\{\min_j \{1/Sec_j\}, 1/|L|\right\}$, then equations (3.48) and (3.49) hold.

Proof.

From equation (3.48), one obtains $\varepsilon \le \dfrac{\sum_{k \in K_j} \sum_{l \in L} Y_{ijkl}}{Sec_j} + 1 - P_{ij}$ for ($i \in I, j \in J$)

Case 1. Suppose teacher *i* teaches course *j*, thus $\sum_{k\in K_j}\sum_{l\in L} Y_{ijkl} \geq 1$ as well as

$$\varepsilon < \frac{1}{Sec_j} \leq \frac{\sum_{k\in K_j}\sum_{l\in L} Y_{ijkl}}{Sec_j} \leq \frac{\sum_{k\in K_j}\sum_{l\in L} Y_{ijkl}}{Sec_j} + 1 - P_{ij}$$

Case 2. Suppose teacher *i* does not teach course *j*, thus $P_{ij} = \sum_{k\in K_j}\sum_{l\in L} Y_{ijkl} = 0$ as well as

$$\varepsilon < 1 = \frac{\sum_{k\in K_j}\sum_{l\in L} Y_{ijkl}}{S_j} + 1 - P_{ij}$$

By referring to equation (3.48), $(\varepsilon + P_{ij} - 1) \leq \frac{\sum_{k\in K_j}\sum_{l\in L} Y_{ijkl}}{Sec_j} \leq P_{ij}$, the value of

$$P_{ij} = \left\lceil \frac{\sum_{k\in K_j}\sum_{l\in L} Y_{ijkl}}{Sec_j} \right\rceil, \left\lceil \frac{\sum_{k\in K_j}\sum_{l\in L} Y_{ijkl}}{Sec_j} \right\rceil + 1, \left\lceil \frac{\sum_{k\in K_j}\sum_{l\in L} Y_{ijkl}}{Sec_j} \right\rceil + 2, \ldots$$ By contradiction, if

$$P_{ij} \geq \left\lceil \frac{\sum_{k\in K_j}\sum_{l\in L} Y_{ijkl}}{Sec_j} \right\rceil + 1, \text{ then } \frac{\sum_{k\in K_j}\sum_{l\in L} Y_{ijkl}}{Sec_j} \geq \varepsilon + P_{ij} - 1 \geq \varepsilon + \left\lceil \frac{\sum_{k\in K_j}\sum_{l\in L} Y_{ijkl}}{Sec_j} \right\rceil > \left\lceil \frac{\sum_{k\in K_j}\sum_{l\in L} Y_{ijkl}}{Sec_j} \right\rceil$$

From equation (3.49), one obtains $\varepsilon \leq \frac{L_i}{|L|} + 1 - V_i$ for ($i \in I$). Let $\frac{1}{L_i}(1 - L_i) \leq 1 - L_i \leq 1 - V_i$, then

$$\varepsilon < \frac{1}{|L|} \leq \frac{L_i}{|L|} + 1 - V_i .$$

By referring to equation (3.49), $(\varepsilon + V_i - 1) \leq \frac{L_i}{|L|} \leq V_i$, the value of

$$V_i = \left\lceil \frac{Li}{|L|} \right\rceil, \left\lceil \frac{Li}{|L|} \right\rceil + 1, \left\lceil \frac{Li}{|L|} \right\rceil + 2, \ldots$$ By contradiction, if $V_i \geq \left\lceil \frac{Li}{|L|} \right\rceil + 1$, then

$$\frac{Li}{|L|} \geq \varepsilon + V_i - 1 \geq \varepsilon + \left\lceil \frac{Li}{|L|} \right\rceil > \left\lceil \frac{Li}{|L|} \right\rceil$$

Thus, we conclude that $\varepsilon < \min\left\{ \min_j \{1/Sec_j\}, 1/|L| \right\}$

The TACS Model III can be represented as follows:

[TACS′ Model III]

Maximize *Objective function* (3.22)

subject to:

Constraints (3.23), (3.25) – (3.34), (3.36) – (3.49)

The maximum number of decision variables and constraints in the proposed mathematical model are $(2|I||J||K||L||M| + |I||J||K||L| + |I||J|+2|I|)$ and $(3|I||J||K||L||M| + 2|I||J||K||L| + |I||J||L||M| + |I||L||M| + |I||J| + 3|J||K| + |I||L| + |J||L| + |L||M| + 4|I| + 3|J|)$, respectively.

3.4.3 Computational Results

To test the performance of the proposed model, it is implemented in ILOG OPL Studio 4.2 on a 2.6GHz Pentium IV PC with 512MB RAM that runs in Microsoft Windows XP operating system. Several data sets are generated in such a way that the data sets correspond to differing values of several parameters. Two different types of randomly generated data sets that represent the TACS problem of varying difficulties are generated. Here, the various parameters are the number of teachers $|I|$, the number of courses $|J|$, the number of sections for each course $|K|$, the maximum number of courses taught N and the number of classrooms available C. The maximum computation time is limited to 24 hours for all the data sets.

The number of days per week is assumed to be five days and the number of time periods per day is eight periods except for problem type 5×5. Each data set consists of five randomly generated data instances. Two different types of data sets are generated and they are known as Group I and Group II data sets, respectively. For Group I data sets, the number of sections for each course is set to a fixed number, while the number of sections for each course varies in the Group II data sets. Tables 3.11 and 3.12 summarize the characteristics of each data set. For all

the data sets, we set the minimum and maximum number of teachers who can teach a particular course (LT_j and UT_j) as shown in Table 3.13.

Table 3.11 Characteristics of Group I data sets

Data set	Number of teachers $\|I\|$	Number of courses $\|J\|$	Number of sections $\|K\|$	Number of days $\|L\|$	Number of time periods per day $\|M\|$	Maximum number of courses taught N	Number of classrooms available C
5×5_1	5	5	2	5	4	1	4
5×5_2	5	5	2	5	4	2	4
10×10_1	10	10	2	5	8	1	4
10×10_2	10	10	2	5	8	2	4
15×15_1	15	15	2	5	8	1	6
15×15_2	15	15	2	5	8	2	6
20×20_1	20	20	2	5	8	1	8
20×20_2	20	20	2	5	8	2	8

Table 3.12 Characteristics of Group II data sets

Data set	Number of teachers $\|I\|$	Number of courses $\|J\|$	Minimum number of sections $\|K_j\|$	Maximum number of sections $\|K_j\|$	Number of days $\|L\|$	Number of time periods per day $\|M\|$	Maximum number of courses taught N	Number of classrooms available C
10×20_1	10	20	2	3	5	8	4	10
10×20_2	10	20	2	4	5	8	4	10
20×30_1	20	30	2	3	5	8	3	15
20×30_2	20	30	2	4	5	8	3	15
20×40_1	20	40	2	3	5	8	4	15
20×40_2	20	40	2	4	5	8	4	15
30×60_1	30	60	2	3	5	8	4	20
30×60_2	30	60	2	4	5	8	4	20

Table 3.13 Minimum and maximum number of teachers for each course

Number of sections	Minimum number of teachers (LT)	Maximum number of teachers (UT)
1	1	1
2	1	2
3	1	2
4	2	3

The maximum size of problems solvable within reasonable time is rather small due to the huge number of constraints and decision variables involved. Tables 3.14 and 3.15 summarize the average best known/optimal objective function values obtained and the average CPU time required to obtain the solutions for Group I and Group II data sets, respectively.

From Table 3.14, we observe that the average CPU time required to obtain the solution increases rapidly when the maximum number of courses taught increases from one to two courses. The average objective function value of the optimal solutions is also increased when the maximum number of courses taught is increased. This is because the chances for each teacher to be assigned the courses and time periods preferred will be higher and each course can be taught by more than one teacher. Similar observations can be found from the results in Table 3.15.

The best known/optimal solution can be found for data sets with the number of teachers = 10 within a few minutes. For 20 teachers, we observed that the CPU time increases rapidly as shown in Table 3.15. The solutions can only be found within 3 to 10 hours. However, the TACS′ Model III could not be solved to optimality for data sets 20×40_1, 20×40_2, 30×60_1 and 30×60_2 within 24 hours. It could be that the problem is too large for the search space to be explored exhaustively. As such, we only report the best known solutions that could be obtained within this computation time limit for these data sets. For 30×60_2 data set, we can only get the best known solutions of two instances. This indicates that the computation time required to find an optimal solution to the TACS problem becomes prohibitively large when the problem size increases.

Table 3.14 Computational results of Group I data sets by OPL Solver

Data set	Objective function value	CPU time (seconds)	Average objective function value	Average CPU time (seconds)
5×5_1(1)	1130	2.43	1052	2.03
5×5_1(2)	1090	2.23		
5×5_1(3)	1040	1.65		
5×5_1(4)	1020	2.04		
5×5_1(5)	980	1.82		
5×5_2(1)	1280	1.85	1216	2.17
5×5_2(2)	1290	2.45		
5×5_2(3)	1110	2.25		
5×5_2(4)	1190	2.03		
5×5_2(5)	1210	2.25		
10×10_1(1)	2340	31.07	2184	18.47
10×10_1(2)	2260	11.59		
10×10_1(3)	2200	22.15		
10×10_1(4)	2020	12.50		
10×10_1(5)	2100	15.03		
10×10_2(1)	2860	49.97	2712	31.79
10×10_2(2)	2780	32.29		
10×10_2(3)	2780	19.70		
10×10_2(4)	2540	10.96		
10×10_2(5)	2600	43.06		
15×15_1(1)	3080	67.23	3170	136.93
15×15_1(2)	3270	200.32		
15×15_1(3)	3130	222.63		
15×15_1(4)	3280	70.53		
15×15_1(5)	3090	123.95		
15×15_2(1)	4170	265.11	4166	431.94
15×15_2(2)	4260	554.36		
15×15_2(3)	4150	189.92		
15×15_2(4)	4320	401.42		
15×15_2(5)	3930	748.90		
20×20_1(1)	4290	583.82	4368	337.98
20×20_1(2)	4330	177.10		
20×20_1(3)	4340	444.72		
20×20_1(4)	4540	172.26		
20×20_1(5)	4340	312.01		
20×20_2(1)	5660	6637.76	5774	4212.92
20×20_2(2)	5730	2127.28		
20×20_2(3)	5770	9367.78		
20×20_2(4)	5880	379.04		
20×20_2(5)	5680	2552.72		

Table 3.15 Computational results of Group II data sets by OPL Solver

Data set	Objective function value	CPU time (seconds)	Average objective function value	Average CPU time (seconds)
10×20_1(1)	7590	1578.91	7766	1183.11
10×20_1(2)	8210	588.28		
10×20_1(3)	7670	311.59		
10×20_1(4)	7560	2789.25		
10×20_1(5)	7800	647.54		
10×20_2(1)	7660	1114.60	7934	1273.15
10×20_2(2)	8480	423.92		
10×20_2(3)	7970	541.53		
10×20_2(4)	7730	3473.21		
10×20_2(5)	7830	812.51		
20×30_1(1)	11010	10283.73	10952	11640.27
20×30_1(2)	11140	9265.63		
20×30_1(3)	10430	10598.13		
20×30_1(4)	10920	18584.29		
20×30_1(5)	11260	9469.55		
20×30_2(1)	13090	17184.70	13468	36019.99
20×30_2(2)	12880	55032.51		
20×30_2(3)	13660	47060.92		
20×30_2(4)	14260	38239.02		
20×30_2(5)	13450	22582.79		
20×40_1(1)	13210[a]	-[b]	13742	-[b]
20×40_1(2)	14010[a]	-[b]		
20×40_1(3)	13360[a]	-[b]		
20×40_1(4)	13860[a]	-[b]		
20×40_1(5)	14270[a]	-[b]		
20×40_2(1)	16190[a]	-[b]	16852	-[b]
20×40_2(2)	17440[a]	-[b]		
20×40_2(3)	17240[a]	-[b]		
20×40_2(4)	16700[a]	-[b]		
20×40_2(5)	16690[a]	-[b]		
30×60_1(1)	20540[a]	-[b]	21074	-[b]
30×60_1(2)	21060[a]	-[b]		
30×60_1(3)	20710[a]	-[b]		
30×60_1(4)	21890[a]	-[b]		
30×60_1(5)	21170[a]	-[b]		
30×60_2(1)	-	-[b]	24830	-[b]
30×60_2(2)	25180[a]	-[b]		
30×60_2(3)	-	-[b]		
30×60_2(4)	-	-[b]		
30×60_2(5)	24480[a]	-[b]		

[a] *The best known solution obtained within 24 hours*
[b] *CPU time = 24 hours*

This difficulty associated with solving the mathematical model primarily stems from the large number of binary variables. This experimentally supports the theoretical results on the NP-completeness of timetabling problems (Even et al., 1976). Consequently, many heuristic algorithms were proposed and developed to deal with the timetabling problem in the literature as mentioned in the previous section. In the following two chapters, several heuristics that can handle large problem sizes would be proposed. Detail comparison among proposed algorithms would also be presented.

3.5 Conclusions

We have proposed three different mathematical programming models for a timetabling problem that combines teacher assignment and course scheduling simultaneously. The first model, TACS Model I, is considered as the basic model which only accommodates some basic or the most common requirements in both problems. This model is extended to two different models, TACS Model II and III, by considering several additional requirements, such as some courses are divided into course sections and so forth. In TACS Model III, we also include several additional requirements which are comparable to those occurring in an engineering faculty of a university in Indonesia, for instance, time periods are considered on a day-hour basis in order to increase teaching flexibility and courses can be conducted at any time period, one course section can only be conducted each day.

We reported the computational results of solving the model for some randomly generated data sets. In TACS Model I, we observe that the average computation time decreases when the maximum load or the number of classrooms available increases. The main reason of this phenomenon is that the constraints related to the maximum number of courses taught or the number of classrooms available can be satisfied easily. On the other hand, the average

computation time increases rapidly when we increase the problem size by adding the number of courses taught.

When we increase the maximum number of courses taught, the average objective value of the optimal solution is also increased, meaning that the chances for each teacher to be assigned more courses and time periods will be higher. Similar conclusions can be drawn from the computational results of TACS Model II.

For the extended TACS Model III, there is difficulty in obtaining the optimal solutions due to the presence of some constraints, such as consecutive constraints and non-linearity constraints, and large number of integer variables as well. As such, in the next chapters, we propose several algorithms to solve the problem especially for large problem sizes.

CHAPTER IV

AN IMPROVEMENT HEURISTIC FOR THE TACS PROBLEM

4.1 Introduction

In the previous chapter, we formulated the university course timetabling problems as mathematical programming models. The proposed basic and extended models combine both teacher assignment and course scheduling problems simultaneously, which causes the entire models to become more complex. It has been shown that the maximum size of problems solvable within reasonable time is rather small due to the huge number of constraints and decision variables involved, especially for the second extended mathematical model (TACS Model III). It would be necessary to develop a heuristic approach in order to find a good solution within a reasonable amount of time. In the next three chapters, we will focus on developing several different heuristics for solving the TACS Model III.

Chapter 4 is started by presenting a simple improvement heuristic for solving the problem. The details of the heuristics are described in the following sub-sections, followed by the computational results and comparisons. Finally, some issues such as the strengths and the weaknesses of the proposed heuristic are summarized in the conclusion part.

4.2 The Proposed Heuristic

Our problem consists of two common sub-problems of the university course timetabling problem: teacher assignment and course scheduling problems. The proposed heuristic will

solve these sub-problems iteratively, and it comprises of three main phases: (1) pre-processing, (2) construction, and (3) improvement (see Figure 4.1).

The first phase is related to the data management. In the second phase, a feasible solution is built. If there is no feasible solution, we need to rearrange the assignments or relax some requirements or constraints. In the improvement phase, the initial solution obtained is further improved. Some examples of heuristic approaches for solving the course scheduling problem were proposed by Loo et al. (1985), Aubin and Ferland (1989) and Wright (1996). We briefly describe each phase below. The details of the proposed heuristic are presented in Appendix *A*.

Pre-Processing Phase

Generate two additional sets, I_j and LM_i

Construction Phase

Allocate teachers to courses and course sections

Allocate course sections to time periods

Improvement Phase

Reallocate teachers to courses and course sections followed by reallocating to time periods

Figure 4.1 Improvement heuristic

4.2.1 The Pre-Processing Phase

The purpose of the pre-processing phase is to construct two additional sets, I_j and LM_i, where $I_j = \left\{i_1^j, i_2^j, ..., i_{|I_j|}^j\right\}$ is the set of teachers who are willing to teach course *j* and sorted in non-increasing order of the PC_{ij} value, and $LM_i = \left\{\left(l^i, m^i\right)^1, \left(l^i, m^i\right)^2, ..., \left(l^i, m^i\right)^{|L||M|}\right\}$ is the set of time periods of teacher *i* which are sorted in non-increasing order of the PT_{ilm} value. For example, $I_1 = \{i^1{}_1, i^1{}_3\}$ is the set of teachers who are willing to teach Course 1 and Teacher 1 has the highest course preference ($PC_{11} \geq PC_{31}$). Similar to LM_i, the first element $(l_i, m_i)^1$

represents a day and a time period with the highest PT_{ilm}^{III} value. The time complexity for these processes are $O(|I|^2|J|)$ and $O(|I||L|^2|M|^2)$, respectively.

4.2.2 The Construction Phase

In this phase, a feasible solution is constructed by accommodating as many course and time period preferences as possible. Each course should be allocated to the teacher who has the highest preference and scheduled to the time periods with the highest time period preferences as well. The entire phase is started with the allocation of teachers to courses and course sections which is related to the teacher assignment problem.

For each course *j*, we refer to set $I_j = \left\{i_1^j, i_2^j, ..., i_{|I_j|}^j\right\}$ in order to find a number of teachers who are going to be allocated to that particular course. The process is started by evaluating Teacher i_1^j with the highest course preference, PC_{ij}. However, in order to generate a feasible initial solution, a *checking procedure* has to be performed. The purpose of this procedure is to ensure the following constraint is satisfied:

- Constraint (3.25): $1 \le \sum_{j \in J} P_{ij} \le N_i \qquad (i \in I)$

  Each teacher cannot teach more than the maximum number of courses taught N_i

When the selected teacher has reached N_i, the next teacher i_2^j with less preference would be considered. This process is continued until the number of selected teachers for course *j* is equal to the minimum requirement, LT_j, as shown in Table 3.13. For example, a course with 3 sections, the minimum number of teacher allocated is 1 teacher, meaning that all sections of that particular course would be initially taught by the selected teacher.

However, it is possible that until the last element $i^{j}_{|I_j|}$, we will not be able to find any teacher from the list I_j. For this case, we can choose a teacher from list I_j who has the lowest number of courses taught by relaxing the requirement of minimum and maximum number of courses taught by each teacher (constraint (3.25)). Therefore, an infeasible initial solution might be generated. Some teachers might have to teach more than the maximum number of courses taught N_i, while other teachers might not teach any course, thereby resulting in infeasibility because of violation of that requirement.

Due to the infeasibility issue in the teacher assignment sub-problem above, we try to improve the initial solution before continuing to the next step. We classify teachers into three different groups:

- $\text{Group}_1 = \left\{ i \in I \,\middle|\, \sum_{j \in J} P_{ij} > N_i \right\}$, this group represents a set of teachers who teach more than the maximum number of courses taught N_i.
- $\text{Group}_2 = \left\{ i \in I \,\middle|\, 1 \le \sum_{j \in J} P_{ij} \le N_i \right\}$, this group represents a set of teachers who teach between the minimum and maximum number of courses taught.
- $\text{Group}_3 = \left\{ i \in I \,\middle|\, \sum_{j \in J} P_{ij} < 1 \right\}$, this group represents a set of teachers who do not teach any course.

The number of teachers in group s is represented as $|Group_s|$, for s = 1, 2 or 3. Our aim is to eliminate teachers from Group 1 as well as to allocate at least one course to teachers in Group 3 such that $|Group_3| = |Group_1| = 0$. For each teacher $i \in \text{Group}_1$, by referring to all courses taught by teacher i, we find another teacher i' who can take over one or more courses taught by teacher i until the total number of courses taught by teacher i is less than or equal to the

maximum number of courses taught ($\sum_j P_{ij} \le N_i$). We need to ensure that teacher *i*' currently has the least number of courses taught.

The following figure illustrates this process. Assuming the minimum and maximum numbers of courses taught by each teacher are 1 and 3 courses, respectively, we refer to teachers in $Group_1$ and select one teacher randomly. For example, Teacher1 currently teaches four courses: Course1, Course2, Course4 and Course6. Since Teacher1 has exceeded the maximum number of courses taught, another teacher has to be found in order to teach one course from Teacher1.

By referring to sets I_1, I_2, I_4 and I_6, we determine which teachers are able to teach courses taught by Teacher1. We have to consider the number of courses taught by those teachers. For example, Course1 can be taught by Teacher2 and Teacher3. Since Teacher2 currently teaches only one course, we decide to choose Teacher2 for teaching Course1 (including all sections of Course1).

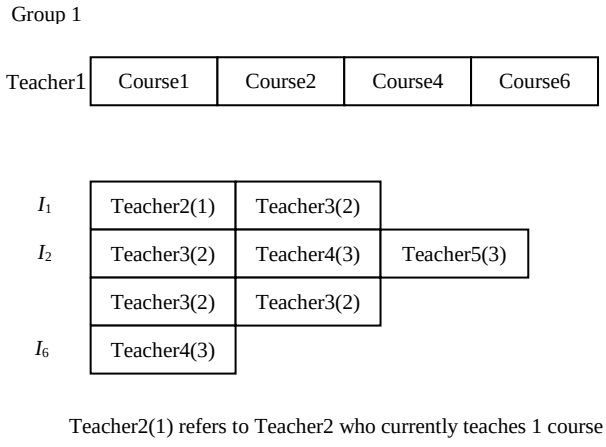


Figure 4.2 A numerical example to illustrate the construction phase

A similar idea is used for eliminating teachers in $Group_3$. We select teacher *i* randomly from $Group_3$. By referring to the list of courses preferred by teacher *i*, J_i, we find a course that is

currently taught by another teacher who has the highest number of courses taught. The selected course is then allocated to teacher *i*. For example, Teacher1 who is currently in Group 3 prefers to teach Course1 and Course2. In the current solution, both courses are taught by Teacher2 and Teacher3, respectively. Assuming that Teacher2 currently teaches 2 courses and Teacher3 teaches 3 courses, we decide to allocate Teacher1 to Course2 and replace Teacher3. These additional steps are performed until $|\text{Group}_1| = |\text{Group}_3| = 0$ or the total number of iterations reaches the preset maximum number of iterations.

The complexity of the above process is $O(|I|^2|J|)$. It is possible to redesign the procedure and make it more efficient, such as by ignoring the maximum and minimum number of courses taught constraints of equation (3.25) in the *Checking procedure*. For this case, the complexity can be reduced to $O(|I||J|)$. However, the possibility of obtaining an infeasible solution might be higher. It would require some additional efforts, such as increasing the number of iterations or increasing the computation time in the improvement phase, in order to achieve better solutions. For special cases such as when the number of sections for each course is set to 1, the complexity will be automatically reduced to $O(|I||J|)$.

For the initial allocation of time periods, each teacher has the time periods sorted based on his/her preferences, LM_i. The number of time periods allocated to each teacher is dependent on the number of courses and course sections allocated during the teacher assignment phase. The idea of the proposed algorithm is almost similar to the teacher assignment phase. We allocate the time periods to teachers based on their time preferences.

For each course *j* section *k* taught by teacher *i*, we choose the first element $(l_i, m_i)^1$ from LM_i as the starting time period. The *Checking procedure* has to be invoked in order to address the feasibility issues. This procedure ensures that the following constraints are satisfied:

- Constraint (3.23): $\sum_{i \in I} \sum_{k \in K_j} X_{ijklm} \leq 1 \qquad (j \in J, l \in L, m \in M)$

  For any course *j*, at most one section can be taught at time period *m*.

- Constraint (3.28): $\sum_{i \in I} \sum_{k \in K_j} Y_{ijkl} \leq 1 \qquad (j \in J, l \in L)$

  For any course *j*, at most one section can be conducted on day *l*.

- Constraint (3.29): $\sum_{j \in J} \sum_{k \in K_j} X_{ijklm} \leq 1 \qquad (i \in I, l \in L, m \in M)$

  Teacher *i* can only teach at most one course section on day *l* time periods *m*.

- Constraint (3.30): $\sum_{i \in I} \sum_{j \in J} \sum_{k \in K} X_{ijklm} \leq C_{lm}^{\mathrm{III}} \qquad (l \in L, m \in M)$

  The number of course sections taught on day *l* time period *m* cannot exceed the number of classrooms available, C_{lm}^{III}.

- Constraint (3.36): $\sum_{j \in J} \sum_{k \in K_j} Y_{ijkl} \leq V_i \qquad (i \in I, l \in L)$

  The number of course sections taught by teacher *i* on day *l* has to be less than or equal to the maximum number of course sections taught by teacher *i* per day, V_i.

When the selected time period is not feasible, the next element $(l_i, m_i)^2$ in LM_i with less preference value would be considered. A similar situation with the teacher assignment sub-problem will be faced if until the last element $(l_i, m_i)^{|L||M|}$ in the list LM_i, the course section has not been allocated yet. For this situation, we have to relax some constraints, such as constraints (3.28), (3.30) and (3.36), and we try to find a set of day and time period from LM_i.

Due to the relaxed constraints, it turns out that some courses might be conducted more than once on the same day, or some time periods have the number of course sections taught that exceed the capacity, or some teachers do not have evenly spread out schedules. These courses, time periods and teachers are kept in Excess Lists 1, 2 and 3, respectively. The number of members in an Excess List *e* is represented as $|EL_e|$, for e = 1, 2 or 3.

- $EL_1 = \left\{ j \in J, l \in L \middle| \sum_{i \in I} \sum_{k \in K_j} Y_{ijkl} > 1 \right\}$, this group represents a set of courses (including their scheduled days) scheduled more than once on a particular day.
- $EL_2 = \left\{ l \in L, m \in M \middle| \sum_{i \in I} \sum_{j \in J} \sum_{k \in K_j} X_{ijklm} > C_{lm}^{\mathrm{III}} \right\}$, this group represents a set of days and time periods where the number of course sections scheduled on that particular day and time period is exceeded the maximum capacity.
- $EL_3 = \left\{ i \in I, l \in L \middle| \sum_{j \in J} \sum_{k \in K_j} Y_{ijkl} > V_i \right\}$, this group represents a set of teachers (including the scheduled days) who teach more than the maximum number of course sections taught per day V_i.

In order to deal with these infeasibility issues, the solution could be improved before continuing to the next phase. This is achieved by reallocating some courses and course sections to new time periods without violating constraints (3.23), (3.28), (3.29), (3.30) and (3.36). For example:

- In Group EL_1, suppose two course sections of Course1 are scheduled on the same day, we select one course section randomly and try to find another different day for that particular course section. However, we need to ensure these constraints (3.23), (3.28), (3.29), (3.30) and (3.36) (3.28), (3.29), (3.30) and (3.36) are not violated.
- In Group EL_2, suppose the number of course sections allocated is greater than the maximum capacity on Day1-Time Period2, we have to remove a certain number of course sections in order to ensure the capacity constraints is not violated. Some course sections are selected randomly and reallocated to other days and time periods.

- In Group EL_3, suppose Teacher1 teaches more than the maximum number of course sections taught per day V_i, we reallocate some course sections taught by Teacher1 to different days and time periods by ensuring there is no violation to other constraints.

These additional steps are performed until $|EC_1| = |EC_2| = |EC_3| = 0$ or the total number of iterations reaches the preset maximum number of iterations. It is important to note that an infeasible initial solution might still occur. However, with the additional steps, the chances of infeasibility would be less. The worst case time complexity of this course scheduling process is $O(|I||J||K||L||M|)$. Again, this time complexity can be reduced through efficient procedures or for some special cases.

After performing the entire processes described above, the total objective function value is then calculated. This objective function is slightly different with equation (3.22) due to additional penalty values, *PENALTY*1 and *PENALTY*2. Here, the terms *PENALTY*1 and *PENALTY*2 reflect the penalty values for the violations of the respective requirements. The details of these calculations are explained as follows:

Course objective function value

$$= \sum_{i\in I}\sum_{j\in J} PC_{ij} \times P_{ij} + \left[\left|Group_1\right| + \left|Group_3\right|\right] \times PENALTY1 \tag{4.1}$$

Time objective function value

$$= \sum_{i\in I}\sum_{j\in J}\sum_{k\in K_j}\sum_{l\in L}\sum_{m\in M} PT^{\mathrm{III}}_{ilm} \times X_{ijklm} + \left[\left|EL_1\right| + \left|EL_2\right| + \left|EL_3\right|\right] \times PENALTY2 \tag{4.2}$$

Total objective function value

= Course objective function value + Time objective function value (4.3)

4.2.3 The Improvement Phase

In this phase, after an initial solution is obtained from the construction phase, two operations are performed in order to seek further improvement. These two operations are reallocation of teachers to courses and course sections, followed by rescheduling course sections to days and time periods. The initial solution is treated as a starting solution in the improvement phase.

The first operation is started by randomly choosing course *j* that is currently taught by teacher i_1, followed by considering another new teacher $i_2 \neq i_1$ $\left(i_2 \in I_j\right)$ without violating the maximum number of courses taught by teacher i_2, N_{i_2}. Two possible moves will be considered as shown in Figure 4.3. The two possible moves considered are to choose either teacher i_2 to be added to the list of teachers who teach course *j* and take over some of the course sections that are currently taught by teacher i_1 (1^{st} move), or teacher i_2 will fully replace teacher i_1 on course *j* (2^{nd} move).

However, if the number of teachers who teach the selected course reaches the maximum number of teachers allowed (UT_j), we can only select the 2^{nd} move; otherwise, a move is chosen randomly. Suppose teacher i_2 could not be found, the process would be terminated and returned to the first operation.

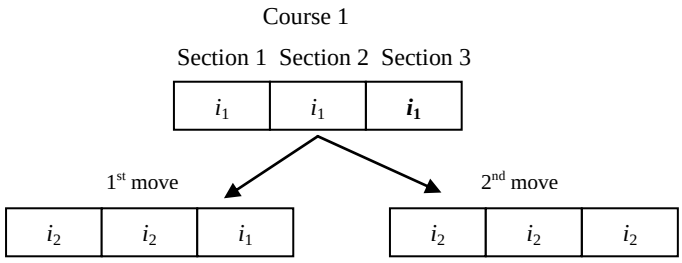


Figure 4.3 Two possible moves

Let K_{i_2} be the set of sections of course *j* taken over by teacher i_2. In the second operation, we check whether it is possible to allocate teacher i_2 to the previous day and time periods scheduled for teacher i_1 to teach course *j* section *k*, where $k \in K_{i_2}$. Otherwise, a new set of days and time periods without constraint violation has to be found.

We present a numerical example to illustrate the neighborhood moves of the proposed algorithm. The first operation is started by choosing one course randomly. Assuming that Course1 with two course sections, Section1 and Section2, has been chosen and both sections are initially taught by Teacher1, we refer to Set I_1 which consists of teachers who are willing to teach Course1. By considering other constraints, such as the minimum and the maximum number of courses taught by each teacher (constraint (3.25)), we examine teachers in Set I_1 and find which teachers would be able to teach Course1.

Let assume I_1 = {Teacher1, Teacher2, Teacher3}, some possible neighborhood moves are either Teacher2 or Teacher3 takes one section or both sections of Course1. Assuming that Teacher2 has higher preference to Course1, we start to evaluate Teacher2. If there is no constraint violation by selecting Teacher2, we choose randomly whether either one course section is allocated to Teacher2 or both sections are allocated to Teacher2. Otherwise, we continue to evaluate Teacher3.

If we can find another teacher (either Teacher2 or Teacher3) who can replace Teacher1, the process is continued to the second operation: rescheduling the changes into a different day or time periods. We check whether it is possible to allocate the new teacher to the previous day and time periods scheduled for Teacher1 to teach that particular course section. Otherwise, a new set of day - time periods with the highest time preference value and without constraint violation has to be found.

Finally, we evaluate the change by calculating the difference of the objective function value of the new timetable and the previous timetable. If the new timetable provides a better total objective function value, we treat the new timetable as the current solution. Otherwise, the process will return to the previous starting solution. These two operations are performed until the total number of iterations reaches the preset maximum number of iterations. Finally, the solution of the problem is the best solution obtained so far. In general, the time complexity of the neighborhood exploration is $O(|I||J||K||L||M|)$.

4.3 Computational Results

To evaluate the performance of the improvement heuristics, we compare the solutions obtained with the solutions obtained by solving the mathematical programming model. The data sets presented in Chapter 3 were used in our computational experiments on the heuristics. The entire heuristics were coded in C++ and tested on the Intel Pentium IV 2.6 GHz PC with 512 MB RAM under the Microsoft Windows XP Operating System.

The values of the parameters used by the improvement heuristic are summarized in Table 4.1. Preliminary experimentation was performed to determine suitable values for the parameters of the proposed heuristic. These values were chosen to ensure a compromise between the computation time and the solution quality.

Table 4.1 Parameter settings for the improvement heuristic

Parameter	Value
Number of iterations in the Construction Phase (*Teacher Assignment Problem*)	$40\times\|I\|$
Number of iterations in the Construction Phase (*Course Scheduling Problem*)	$20\times\|J\|$
Number of iterations in the Improvement Phase	$\|I\|\times\|J\|\times\|L\|\times\|M\|$
*PENALTY*1, *PENALTY*2	1,000

Table 4.2 Comparison of the heuristic results and the optimal solutions on Group I data sets

Data Set	Solution obtained by commercial software		The improvement heuristic				
	Objective function value	CPU time (seconds)	Average objective function value	Best objective function value	Average CPU time (seconds)	$\Phi_{(1)}^{heuristic}$ (%)	$\Phi_{(2)}^{heuristic}$ (%)
5×5_1(1)	1130	2.43	1100	1100	0.00	2.65	2.65
5×5_1(2)	1090	2.23	820	910	0.00	24.77	16.51
5×5_1(3)	1040	1.65	810	880	0.02	22.12	15.38
5×5_1(4)	1020	2.04	800	800	0.02	21.57	21.57
5×5_1(5)	980	1.82	910	930	0.00	7.14	5.10
5×5_2(1)	1280	1.85	1100	1140	0.00	14.06	10.94
5×5_2(2)	1290	2.45	1020	1020	0.02	20.93	20.93
5×5_2(3)	1110	2.25	910	910	0.02	18.02	18.02
5×5_2(4)	1190	2.03	810	850	0.02	31.93	28.57
5×5_2(5)	1210	2.25	1060	1110	0.02	12.40	8.26
10×10_1(1)	2340	31.07	2180	2230	0.08	6.84	4.70
10×10_1(2)	2260	11.59	1610	1800	0.08	28.76	20.35
10×10_1(3)	2200	22.15	1970	1980	0.06	10.45	10.00
10×10_1(4)	2020	12.50	1720	1780	0.08	14.85	11.88
10×10_1(5)	2100	15.03	1880	1880	0.06	10.48	10.48
10×10_2(1)	2860	49.97	2390	2650	0.22	16.43	7.34
10×10_2(2)	2780	32.29	2070	2320	0.23	25.54	16.55
10×10_2(3)	2780	19.70	2090	2510	0.30	24.82	9.71
10×10_2(4)	2540	10.96	1950	2180	0.24	23.23	14.17
10×10_2(5)	2600	46.03	2060	2300	0.25	20.77	11.54
15×15_1(1)	3080	67.23	2790	2940	0.20	9.42	4.55
15×15_1(2)	3270	200.32	3100	3120	0.20	5.20	4.59
15×15_1(3)	3130	222.63	2710	2820	0.22	13.42	9.90
15×15_1(4)	3280	70.53	2910	3000	0.23	11.27	8.54
15×15_1(5)	3090	123.95	2580	2700	0.20	16.50	12.62
15×15_2(1)	4170	265.11	3920	3920	0.94	6.00	6.00
15×15_2(2)	4260	554.36	3930	3930	0.97	7.75	7.75
15×15_2(3)	4150	189.92	3850	3850	0.88	7.23	7.23
15×15_2(4)	4320	401.42	3980	3980	0.89	7.87	7.87
15×15_2(5)	3930	748.90	3560	3560	0.97	9.41	9.41
20×20_1(1)	4290	583.82	4210	4210	0.56	1.86	1.86
20×20_1(2)	4330	177.10	3740	3740	0.52	13.63	13.63
20×20_1(3)	4340	444.72	3900	3900	0.56	10.40	10.14
20×20_1(4)	4540	172.26	4420	4420	0.59	2.64	2.64
20×20_1(5)	4340	312.01	4000	4000	0.52	7.83	7.83
20×20_2(1)	5660	6637.76	5440	5440	2.42	3.89	3.89
20×20_2(2)	5730	2127.28	4970	4970	2.23	13.26	13.26
20×20_2(3)	5770	9367.78	5060	5060	2.19	12.31	12.31
20×20_2(4)	5880	379.04	5540	5540	2.19	5.78	5.78
20×20_2(5)	5680	2552.72	5230	5230	2.17	7.92	7.92
					Average	13.28	10.56
					Maximum	31.93	28.57
					Minimum	1.86	1.86

Table 4.3 Comparison of the heuristic results and the optimal solutions on Group II data sets

Data Set	Solution obtained by commercial software		The improvement heuristic				
	Objective function value	CPU time (seconds)	Average objective function value	Best objective function value	Average CPU time (seconds)	$\Phi_{(1)}^{heuristic}$ (%)	$\Phi_{(2)}^{heuristic}$ (%)
10×20_1(1)	7590	1578.91	6040	6520	0.95	20.42	14.10
10×20_1(2)	8210	588.28	6740	5920	1.52	17.90	15.71
10×20_1(3)	7670	311.59	6130	6470	1.05	20.08	15.65
10×20_1(4)	7560	2789.25	6270	6240	1.02	17.06	17.46
10×20_1(5)	7800	647.54	6060	6910	1.00	22.31	11.41
10×20_2(1)	7660	11145.60	5960	6590	1.09	22.19	13.97
10×20_2(2)	8480	423.92	7130	7080	1.13	15.92	16.51
10×20_2(3)	7970	541.53	6340	6290	1.13	20.45	21.08
10×20_2(4)	7730	3473.21	6400	6580	1.16	17.21	14.88
10×20_2(5)	7830	812.51	6140	6370	0.89	21.58	18.65
20×30_1(1)	11010	10283.73	8420	9740	7.88	23.52	11.53
20×30_1(2)	11140	9265.63	8250	9480	8.00	25.94	14.90
20×30_1(3)	10430	10598.13	7750	9370	8.14	25.70	10.16
20×30_1(4)	10920	18584.29	7960	9260	9.09	27.11	15.20
20×30_1(5)	11260	9469.55	8550	9730	8.48	24.07	13.59
20×30_2(1)	13090	17184.70	10280	11420	15.55	21.47	12.76
20×30_2(2)	12880	55032.51	10010	10890	16.48	22.28	15.45
20×30_2(3)	13660	47060.92	10070	11430	16.58	26.28	16.33
20×30_2(4)	14260	38239.02	10070	11820	17.47	29.38	17.11
20×30_2(5)	13450	22582.79	10800	11060	18.39	19.70	17.77
20×40_1(1)	13210[a]	-[b]	9970	11890	16.24	24.53	9.99
20×40_1(2)	14010[a]	-[b]	10490	11740	16.28	25.12	16.20
20×40_1(3)	13360[a]	-[b]	10230	11820	17.03	23.43	11.53
20×40_1(4)	13860[a]	-[b]	10530	11950	16.33	24.03	13.78
20×40_1(5)	14270[a]	-[b]	10480	11720	16.61	26.56	17.87
20×40_2(1)	16190[a]	-[b]	13200	14170	48.14	18.47	12.48
20×40_2(2)	17440[a]	-[b]	13500	13990	50.03	22.59	19.78
20×40_2(3)	17240[a]	-[b]	12630	14350	48.55	26.74	16.76
20×40_2(4)	16700[a]	-[b]	12240	13960	50.44	26.71	16.41
20×40_2(5)	16690[a]	-[b]	12830	13560	46.89	23.13	18.75
30×60_2(1)	20540[a]	-[b]	15810	17990	217.37	23.03	12.41
30×60_2(2)	21060[a]	-[b]	15760	18320	204.58	25.17	13.01
30×60_2(3)	20710[a]	-[b]	16050	18530	218.55	22.50	10.53
30×60_2(4)	21890[a]	-[b]	16580	19260	203.39	24.26	12.01
30×60_2(5)	21170[a]	-[b]	16430	18420	196.41	22.39	12.99
30×60_1(1)	-	-[b]	18940	21290	377.36	-	-
30×60_1(2)	25180[a]	-[b]	19260	20700	397.39	23.51	17.79
30×60_1(3)	-	-[b]	19030	20390	420.00	-	-
30×60_1(4)	-	-[b]	18810	21820	396.88	-	-
30×60_1(5)	24480[a]	-[b]	18770	20710	373.61	23.33	15.40
					Average	22.87	14.92
					Maximum	29.38	21.08
					Minimum	15.92	9.99

[a] *The best known solution obtained within 24 hours*
[b] *CPU time = 24 hours*

For each data set, the proposed algorithm was executed 20 times with different random seeds. Tables 4.2 and 4.3 summarize the results obtained from the proposed heuristic for Group I and

Group II data sets, respectively. As described in earlier chapters, the TACS' Model III was initially solved by commercial software (ILOG OPL Studio 4.2). Unfortunately, the optimal solution for data sets 20×40_1, 20×40_2, 30×60_1 and 30×60_2 could not be computed within the time limit of 24 hours. Thus, only the best known solutions for those data sets were reported. These numerical results indicate that the computing time required to find an optimal solution to the problem becomes prohibitively large when the problem size increases.

A comparison of the objective function values and computation times (in seconds) obtained by the heuristic and the integer programming model was presented. The comparison is done by calculating the percentage deviation of the best objective function value of the proposed algorithm A ($Z_{\text{best}}^{\text{A}}$) as well as the average objective function values of the proposed algorithm A ($\overline{Z}^{\text{A}}$) from the best known/optimal value (Z^*) as follows:

$$\Phi_{(1)}^{\text{A}} = 100 \times \left(\frac{Z^* - \overline{Z}^{\text{A}}}{Z^*} \right) \tag{4.4}$$

$$\Phi_{(2)}^{\text{A}} = 100 \times \left(\frac{Z^* - Z_{\text{Best}}^{\text{A}}}{Z^*} \right) \tag{4.5}$$

For Group I data sets, we observe that the proposed heuristic can yield feasible solutions with the deviation of the best and the average objective function values of the proposed algorithm from the best known/optimal value being less than 32% and 29%, respectively. The best or minimum value of parameters $\Phi_{(1)}^{\text{heuristic}}$ and $\Phi_{(2)}^{\text{heuristic}}$ is only 1.86%. The average values of both parameters are 13.28 and 10.56, respectively.

Similar observations are obtained from the results of Group II data sets. The deviation of the best and the average objective function values of the proposed algorithm from the best known/optimal value are not more than 29.38% and 21.08%, respectively. However, the best

values of parameters $\Phi_{(1)}^{\text{heuristic}}$ and $\Phi_{(2)}^{\text{heuristic}}$ are quite high compared with those of Group I data sets. The best or minimum values of parameters $\Phi_{(1)}^{\text{heuristic}}$ and $\Phi_{(2)}^{\text{heuristic}}$ are only 15.92% and 9.99%, respectively. The average values are considered high for both parameters (22.87% and 14.92%).

We notice that the proposed heuristic is able to find solutions within a reasonable amount of computation time for both group data sets although some solutions are not good enough. Some problems that were unsolved by the ILOG OPL Studio software can be solved by the proposed heuristic within reasonable computation time, as can be seen in data sets such as that of 20×40_1, 20×40_2, 30×60_1, and 30×60_2.

Table 4.4 Distribution teachers based on the number of courses taught

Data Set	Maximum number of courses taught, N_i	Average percentage of teachers teaching the following number of courses (%)				Variance
		1	2	3	4	
5×5_1	1	100.0	0.0	0.0	0.0	0.00
5×5_2	2	72.0	28.0	0.0	0.0	0.16
10×10_1	1	100.0	0.0	0.0	0.0	0.00
10×10_2	2	41.3	25.3	0.0	0.0	0.23
15×15_1	1	100.0	0.0	0.0	0.0	0.00
15×15_2	2	64.0	36.0	0.0	0.0	0.22
20×20_1	1	100.0	0.0	0.0	0.0	0.00
20×20_2	2	74.0	26.0	0.0	0.0	0.18
10×20_1	4	8.0	30.0	34.0	28.0	0.84
10×20_2	4	6.0	20.0	30.0	44.0	0.85
20×30_1	3	24.0	46.0	30.0	0.0	0.53
20×30_2	3	20.0	39.0	41.0	0.0	0.56
20×40_1	4	13.0	27.0	35.0	25.0	0.95
20×40_2	4	14.0	15.0	28.0	43.0	1.13
30×60_1	4	9.3	30.7	34.0	26.0	0.88
30×60_2	4	6.7	21.3	35.3	36.7	0.84

Another observation of interest is on the number of courses taught distribution of the teachers with respect to the maximum number of courses taught that we specify (see Table 4.4). For Group I data sets, we notice that the proposed heuristic distributes the number of courses taught evenly to teachers. The variances obtained are reasonably small ($\leq 0.23\%$). However, for Group II data sets, the variances are found to be large with only a small percentage of teachers having very small number of courses taught.

4.4 Conclusions

This chapter has presented an efficient approach to solve a large TACS problem that cannot be easily solved by commercial software. For large problems, it could be difficult to find and prove the existence of an optimal solution, especially within short computation time. The proposed heuristic solves these sub-problems iteratively.

The results obtained from the proposed heuristic were compared against the best known/optimal solutions obtained by the ILOG OPL Studio Software. Although the proposed heuristic is able to yield feasible solutions within reasonable computation time, we notice that the percentage of solution deviation from best known/optimal objective function value is not good enough due to inability of the solution to escape from local optimum.

Though the entire heuristic described here is a simple improvement heuristic, it is worth noting that the procedures in the improvement phase allow for various types of modifications, such as incorporating metaheuristic procedures to the improvement phase. Such suitable hybridization of the heuristic methods may exhibit significantly better performance in terms of solution quality and the time to obtain the solutions, as noted by Purchinger and Raidl (2005).

In the next two chapters, we look into several ways of improving the proposed heuristic to obtain solutions of better quality. This would include the hybridization of the proposed heuristic with other types of heuristics or exact algorithms as mentioned in Chapter 2.

CHAPTER V
HYBRIDIZATION OF HEURISTICS FOR THE TACS PROBLEM

5.1 Introduction

In the previous chapter, we have introduced an improvement heuristic based on a greedy heuristic for solving the TACS Model III. The limitation of the heuristic, which is related to the quality of the solution, was highlighted. In this chapter, we propose another type of heuristic, known as the hybrid algorithm.

The motivation of hybridization is to obtain better solutions compared with those of the improvement heuristic proposed in Chapter 4. Instead of applying a single algorithm, we consider the hybridization of several algorithms that has risen considerably among researchers in combinatorial optimization over the last few years. The best results found for many practical or academic optimization problems are often obtained by hybrid algorithms (Talbi, 2002). In this chapter, we propose three different hybrid algorithms based on hybridization of Simulated Annealing and Tabu Search algorithms.

The rest of this chapter is organized as follows. Section 5.2 discusses three different hybrid algorithms proposed in detail. In Section 5.3, a comparative study on their performance is conducted. The experiment on several random generated data sets was carried out to verify the performance of the proposed algorithms. Finally, some conclusions and directions for possible future research are described in Section 5.4.

5.2 Hybridization of Heuristics

Three different collaborative hybrid algorithms, namely, Algorithms SA1, SA2 and SA-TS, are introduced to solve the TACS Model III. These algorithms are mainly based on Simulated Annealing (SA). SA algorithm is a type of local-search heuristic algorithm to avoid getting trapped at a local minimum by accepting ”Uphill” moves that deteriorate the objective function value, using a probabilistic acceptance criterion (Kirkpatrick et al., 1983). This situation makes it possible for the solution to escape from the local optimum, and finally to converge to the global optimum.

However, SA has some drawbacks, such as completely memoryless, excessive or unnecessary moves during high temperatures. Owing to the random nature of SA, potentially good solutions can often be missed (Lim et al., 2005). In order to minimize these drawbacks, we incorporate other properties of Tabu Search, such as the intensification strategy, aspiration criterion and tabu list.

Intensification strategy is a feature that explores more thoroughly the portions of the search space that seem promising in order to ensure that the best solutions in certain areas are indeed found. This strategy is incorporated in each algorithm proposed. In Algorithm SA-TS, the concept of tabu list for keeping track the last visited movement and avoiding cycles is introduced. Tabu list is added in the acceptance-rejection process (evaluation process) of SA in order to avoid excessive or unnecessary moves especially during high temperatures. When a move belongs to the tabu list at any iteration, it would not be accepted, unless a tabu move passes the aspiration criterion or provides a better objective function value.

Similar to the improvement heuristics proposed in Chapter 4, each algorithm comprises of three main phases: (1) pre-processing, (2) construction, and (3) improvement. Each proposed algorithm combines a greedy heuristic and a hybridization of Simulated Annealing and Tabu Search algorithms sequentially. The greedy heuristic is used for generating an initial solution in the construction phase, while the hybridization of SA and TS is implemented for improving the solution in the improvement phase.

For the first two phases, pre-processing and construction phases, each algorithm follows the same procedures. The difference lies in in the improvement phase that would be explained after the descriptions of pre-processing and construction phases.

5.2.1 The Pre-Processing Phase

This phase is similar to that in Section 4.2.1. The main purpose is to generate two different sets I_j and LM_i. For more details, please refer to Section 4.2.1

5.2.2 The Construction Phase

The construction phase is used primarily to obtain an initial feasible solution. In this phase, the TACS problem is divided into two sub-problems: the teacher assignment and course scheduling problems.

As part of the proposed hybrid algorithms, we propose a mathematical programming model that only focuses on the teacher assignment problem, namely, TA model. This model is a modification of the TACS Model III presented in Chapter 3. In TA model, we only consider the constraints which are related to the teacher assignment problem.

We define the following decision variables:

X'_{ijk} = 1 if teacher *i* teaches course *j* section *k*; 0 otherwise $\left(i \in I, j \in J, k \in K_j\right)$

P'_{ij} = 1 if teacher *i* teaches course *j*; 0 otherwise $\left(i \in I, j \in J\right)$

[TA Model]:

Maximize $Z_{TA} = \sum_{i \in I} \sum_{j \in J} PC_{ij} \times P'_{ij}$ (5.1)

subject to:

$$P'_{ij} = \left\lceil \sum_{k \in K_j} \frac{X'_{ijk}}{Sec_j} \right\rceil \qquad \left(i \in I, j \in J\right) \tag{5.2}$$

$$1 \le \sum_{j \in J} P'_{ij} \le N_i \qquad \left(i \in I\right) \tag{5.3}$$

$$LT_j \le \sum_{i \in I} P'_{ij} \le UT_j \qquad \left(j \in J\right) \tag{5.4}$$

$$X'_{ijk} = 0 \qquad \left(i \in I, j \notin J_i, k \in K_j\right) \tag{5.5}$$

$$\sum_{i \in I} X'_{ijk} = 1 \qquad \left(j \in J, k \in K_j\right) \tag{5.6}$$

$$X'_{ijk} \in \{0,1\} \qquad \left(i \in I, j \in J, k \in K_j\right) \tag{5.7}$$

$$P'_{ij} \in \{0,1\} \qquad \left(i \in I, j \in J\right) \tag{5.8}$$

The objective function in (5.1) only involves the course preference function that needs to be maximized. Equation (5.2) ensures that when teacher *i* teaches at least one section of course *j*, the value of P'_{ij} would be 1, meaning that teacher *i* teaches course *j*. Note that equation (5.2) is nonlinear but it can be linearized by the following equation with sufficiently small but positive ε:

$$Sec_j \times \left(\varepsilon + P'_{ij} - 1\right) \le \sum_{k \in K_j} X'_{ijk} \le Sec_j \times P'_{ij} \qquad \left(i \in I, j \in J\right) \tag{5.9}$$

Equation (5.3) ensures that there is a limit to the number of courses taught by each teacher. Equation (5.4) restricts the number of teachers who could teach a particular course. Equation

(5.5) ensures that teachers will not be assigned courses that they are unable to teach. Equation (5.6) assumes that each course section can only be taught by one teacher. Finally, constraints (5.7) and (5.8) represent the integrality constraints for the decision variables X'_{ijk} and P'_{ij}. The optimal solution obtained Z_{TA} is denoted as *initial_ta*.

Based on some preliminary experiments, the TA Model can be optimally solved by commercial software, ILOG OPL Studio 4.2. However, if this sub-problem cannot be optimally solved especially for large-size problems, the proposed heuristic in the construction phase presented in Chapter 4 can be applied. The optimal solution of the TA model is applied as a part of the initial solution in the proposed hybrid algorithm for the overall TACS problem.

The second sub-problem, which is related to the course scheduling problem, would be solved by a simple greedy heuristic. The idea is similar to that of an improvement heuristic proposed in Chapter 4. An initial feasible solution, *initial_cs*, is constructed by satisfying as much time period preferences as possible. The entire process in the construction phase is briefly outlined in Algorithm 1 (Figure 5.1). The flow chart of Algorithm 1 is presented in Appendix *B*1.

Algorithm 1:

CONSTRUCTION PHASE ( )

(1) Solve TA Model, determining X'_{ijk} and P'_{ij}

(2) Obtain the objective function value of the teacher assignment problem, *initial_ta*

(3) Allocate course *j* section *k* to time periods $\left(\forall j \in J, \forall k \in K_j\right)$

(4) Calculate the objective function value of the course scheduling problem, *initial_cs*

(5) Set the solutions obtained as the initial solution

(6) Calculate the initial objective function value, *initial_sol* = *initial_ta* + *initial_cs*

Figure 5.1 Pseudocode of the construction phase

For illustration, the following figure presents part of the solution obtained for data set 5×5_1(1) in the construction phase.

Day	1					2				
Period	1	2	3	4	5	1	2	3	4	5
Teacher1								4(2)	4(2)	
Teacher2	3(1)	3(1)								
Teacher3		5(1)	5(1)							
Teacher4						2(2)	2(2)			
Teacher5						1(2)	1(2)			

Figure 5.2 Part of the initial solution of data set 5×5_1(1)

As illustrated above, each course section is scheduled on a particular day and a certain number of time periods. For instance, Teacher1 teaches Course4-Section2 on Day2-Time Periods3 and 4. Teacher4 and Teacher5 teach different course sections on the same day and time periods. This initial solution would be further improved in the next phase, the improvement phase.

5.2.3 The Improvement Phase

The improvement phase aims to improve the quality of the solution by applying Simulated Annealing algorithm. SA was originally developed by Kirkpatrick et al. (1983) for finding good solutions to a wide variety of combinatorial optimization problems. It is a type of metaheuristics which avoids getting trapped at a local optimum by accepting deteriorating moves that worsen the objective function value, using a probabilistic acceptance criterion:

$$P(Acceptance/\Delta, T_n) = e^{-\Delta/T_n} \tag{5.10}$$

where Δ is the difference of objective function values between two successive moves and T_n is the temperature at iteration n. The following table links the terminology of the TACS problem to SA terminology.

Table 5.1 Terminology relationship

SA terminology	Problem equivalent
System	Teacher assignment and course scheduling problems
Energy state	Configuration of teachers, courses, course sections, days and time periods
Energy	The objective function value in TACS Model III
Particles	Teachers, courses, course sections, days and time periods

In this study, we use the geometric cooling schedule to update the temperature, which is similar to the one applied by Saleh Elmohamed et al. (1998) and Liu and Ong (2002). This geometric cooling schedule computes the temperature, T_{n+1} at iteration $n+1$ by multiplying the temperature at iteration n, T_n, with the constant cooling factor α:

$$T_{n+1} = \alpha \times T_n \tag{5.11}$$

As mentioned earlier, three hybrid algorithms, namely, Algorithms SA1, SA2 and SA-TS, are proposed. The main difference between pure Simulated Annealing (Kirkpatrick et al., 1983) and our proposed algorithms lies in the additional strategy applied. The intensification strategy, which is originally from Tabu Search, is incorporated to each proposed algorithm. The idea of the intensification strategy is to focus the search once again starting from the best solution obtained in order to further improve the quality of the solutions if there is no improvement of the solution obtained after a certain number of iterations.

The neighborhood structure applied to each algorithm is similar to the one proposed in Chapter 4. Basically, the neighborhood structure is mainly based on two operations: reallocation of teachers to courses and course sections and rescheduling of course sections to days and time periods. Algorithm SA1 only applies the acceptance-rejection process (evaluation process) of SA at the end of two operations. On the other hand, Algorithms SA2 and SA-TS apply the evaluation process at the end of each operation. In Algorithm SA-TS, we also incorporate some features of Tabu Search, such as the aspiration criterion and tabu list. In the following sub-sections, each algorithm is further described in more detail.

5.2.3.1 Algorithm SA1

After obtaining an initial solution from the construction phase, we begin two operations in order to seek further improvements. These two operations include reallocation of teachers to courses and course sections (the first operation), followed by rescheduling course sections to days and time periods (the second operation).

We present a numerical example to illustrate the neighborhood moves of Algorithm SA1. The first operation is started by choosing one course randomly. Assuming that Course1 with two course sections, Section1 and Section2, is chosen and both sections are currently taught by Teacher1, we refer to Set I_1 which consists of teachers who are willing to teach Course1. By considering other constraints, such as the minimum and the maximum number of courses taught by each teacher (constraint (3.25)), we examine teachers in Set I_1 in order to determine which teachers would be able to teach Course1.

Let assume I_1 = {Teacher1, Teacher2, Teacher3}, some possible neighborhood moves are either Teacher2 or Teacher3 teaches one section or both sections of Course1. Since Teacher2 has higher preference to Course1, we evaluate Teacher2. If there is no constraint violation by selecting Teacher2, we choose either one or two sections would be allocated to Teacher2 randomly. Otherwise, we continue to evaluate Teacher3.

If we can find a new teacher (either Teacher2 or Teacher3) who can replace Teacher1, the process is continued to the second operation: rescheduling the changes into a different day or time periods. We evaluate whether it is possible to allocate the new teacher to the previous day and time periods scheduled for Teacher1 to teach that particular course section. Otherwise, a

new set of days and time periods with no constraint violation has to be found. However, if either the first or the second operation is not successful, we simply return to the first operation.

Figure 5.3 describes how we evaluate the neighborhood movements. Let *S*1, *S*2, … *Sx* and *T*1, *T*2, …, *Tv* be the possible neighborhood moves generated in the first and second operations, respectively. Let assume that *S*1 and *T*2 are the selected neighborhood moves of the first and the second operations, respectively, we then evaluate the total objective function value obtained.

If these neighborhood moves can give a better total objective function value, they would be accepted. On the other hand, if the total objective function value is worse than that of the current solution, we evaluate whether the neighborhood moves are either accepted or rejected with a certain probability. In this case, the acceptance-rejection process (the evaluation process) of SA is only applied at the end of two operations.

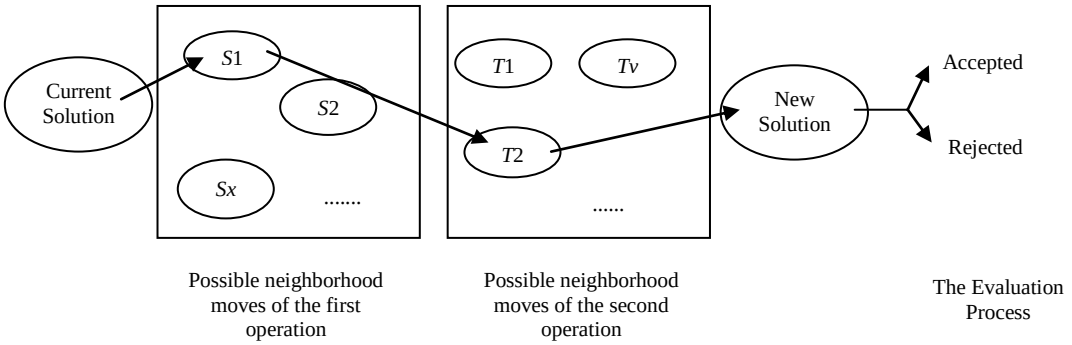


Figure 5.3 Solution representation of Algorithm SA1

The details of SA1 are summarized in Algorithm 2 (Figure 5.4). The entire algorithm will be terminated if the total number of iterations of the outer loop reaches the preset maximum number of iterations, *outer_loop*. The acceptance-rejection process (the evaluation process) is

summarized in Algorithm 3 (Figure 5.5), which is part of Algorithm 2. Flow charts of both algorithms are presented in Appendix *B*2 and *B*3, respectively.

Algorithm 2:
SA1 ( )
(1) Initialize the parameters
(2) Set the best solution, *best_sol* = *initial_sol*
(3) Set the current solution, *current_sol* = *initial_sol*
(4) Set the total number of iterations, *num_iter* = 0
(5) Set the number of non improvement iterations, *no_improv* = 0
(6) **While** the total number of iterations, *num_iter* is less than the preset maximum number of iterations, *outer_loop* **do**:
(7) **Repeat** *inner_loop* times:
(8) Apply the first operation (reallocation teachers to courses and course sections)
(9) **If** the first operation is successful
(10) Apply the second operation (rescheduling course sections to days and time periods)
(11) **If** the second operation is successful
(12) Evaluate the changes by applying **Algorithm 3**
(13) **Else**
(14) Return to Step (8)
(15) **Else**
(16) Return to Step (8)
(17) Update temperature T_{num_iter}
(18) **If** *current_sol* is worse than *best_sol*
(19) *no_improv* := *no_improv* + 1
(20) **If** *no_improv* > the maximum number of non improvement iterations, *limit*
(21) Apply the intensification strategy
(22) *num_iter* := *num_iter* + 1
(23) **Endwhil**e
(24) Report the best solution, *best_sol*

Figure 5.4 Pseudocode of Algorithm SA1

Algorithm 3:
EVALUATION PROCESS SA ( )
(1) Calculate the new solution, *new_sol*
(2) Calculate the change of the objective function, Δ := *new_sol* – *current_sol*
(3) **If** $\Delta > 0$
(4) Update the current solution, *current_sol*
(5) **If** *current_sol* is better than *best_sol*
(6) Update the best solution, *best _sol* = *current_sol*
(7) Else
(8) Choose a random number r_1 uniformly from [0,1]
(9) **If** $r_1 < exp^{\Delta/T_{num\_iter}}$
(10) Accept the new solution, *new_sol*
(11) Update the current solution, *current_sol*
(12) Else
(13) Return to the current solution, *current_sol*

Figure 5.5 Evaluation process of Algorithms SA1 and SA2

5.2.3.2 Algorithm SA2

In this algorithm, two similar operations are involved: reallocating teachers to courses and course sections and rescheduling these changes into other days and time periods. The major difference between Algorithms SA1 and SA2 lies in how we apply the evaluation process (Algorithm 3).

As described in Section 5.2.3.1, the evaluation process is only applied at the end of the two operations in Algorithm SA1. On the other hand, we incorporate the acceptance-rejection process (the evaluation process) after each operation is conducted in Algorithm SA2 (Algorithm 4), as presented in Figure 5.6 and Appendix *B*4. For illustration purpose, please refer to Figure 5.7.

Algorithm 4:
SA2 ( )
(1) Initialize the parameters
(2) Set the best solution, *best_sol* = *initial_sol*
(3) Set the current solution, *current_sol* = *initial_sol*
(4) Set the total number of iterations, *num_iter* = 0
(5) Set the number of non improvement iterations, *no_improv* = 0
(6) **While** the total number of iterations, *num_iter* is less than the preset maximum number of iterations, *outer_loop* **do**:
(7) **Repeat** *inner_loop* times:
(8) Apply the first operation
(9) **If** the first operation is successful
(10) Evaluate the change by applying **Algorithm 3**
(11) Apply the second operation
(12) **If** the second operation is successful
(13) Evaluate the change by applying **Algorithm 3**
(14) **Else**
(15) Return to Step (8)
(16) Update temperature T_{num_iter}
(17) **If** *current_sol* is worse than *best_sol*
(18) *no_improv* := *no_improv* + 1
(19) **If** *no_improv* > the maximum number of non improvement iterations, *limit*
(20) Apply the intensification strategy
(21) *num_iter* := *num_iter* + 1
(22) **Endwhile**
(23) Report the best solution, *best_sol*

Figure 5.6 Pseudocode of Algorithm SA2

Figure 5.7 describes how we apply the evaluation process at the end of each operation. As described earlier, the acceptance-rejection process of SA is applied at the end of each operation. Let *S*1 be the selected neighbor move of the first operation, we continue to evaluate the change of the teacher assignment objective function value by applying the acceptance-rejection process (Algorithm 3). If S1 is accepted, we proceed to the second operation. The selected neighbor move of the second operation is again evaluated in terms of the change of the course scheduling objective function value.

However, if the first operation is rejected, the second operation is still performed by another different strategy. The operation is started by choosing section *k* of course *j* randomly, where $k \in K_j$. This course section will be allocated to other possible time periods. The change of the course scheduling objective function value is then evaluated by the acceptance-rejection process (Algorithm 3).

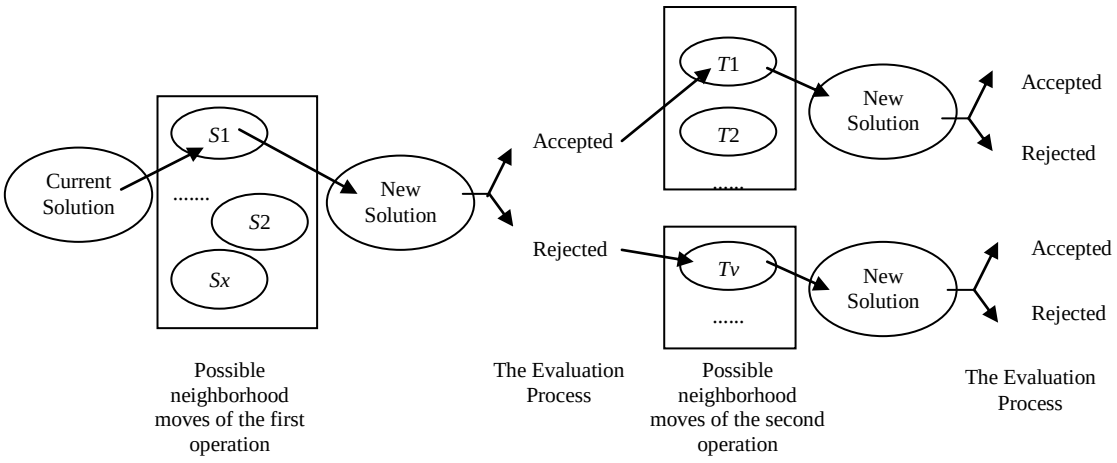


Figure 5.7 Solution representation of Algorithm SA2

5.2.3.3 Algorithm SA-TS

In Algorithm SA-TS, we provide a framework involving the hybridization of Simulated Annealing (SA) and Tabu Search to develop and improve the quality of the solution.

Similar to Algorithm SA2, the phase is started by applying the first operation, the reallocation of teachers to courses and course sections. Several features from Tabu Search, such as tabu list and aspiration criterion, are introduced in this algorithm. Tabu Search explicitly uses a short term memory which is implemented as a tabu list for escaping from local minima and avoiding cycles. A tabu list keeps track of the most recently visited solutions and forbids moves toward them unless a move that satisfies the aspiration criterion even if it is forbidden by tabu conditions. The most commonly used the aspiration criterion is to allow a solution which is better than the best solution obtained so far.

In pure SA algorithm, a deteriorating move would be evaluated by using a probabilistic acceptance criterion, as shown in equation (5.10). In an effort to avoid excessive or unnecessary moves, which will deteriorate the objective function value especially during high temperatures, we include an additional evaluation step after the probabilistic acceptance calculation. When a worse move belongs to the tabu list for a given iteration, it is not allowed to be accepted. On the other hand, a worse non-tabu move can either be accepted or rejected with certain probability.

The way we choose the tabu list depends on the element replaced in the current solution. In the first operation (reallocation teachers to courses or course sections), the tabu list is denoted as *tabu*1, which contains pairs of teacher *i* and course *j* visited in the last *length*1 iterations. For example, in the current solution, Course1 is currently taught by Teacher1. After the first

operation is conducted, a new solution is generated by allocating Teacher2 to Course1. In this case, a pair of Teacher1 and Course1 is included in the tabu list, *tabu*1.We are not allowed to choose this pair in the last *length*1 iterations unless it satisfies the aspiration criterion or gives a better objective function value.

The improvement phase is then continued to the second operation. We introduce another tabu list for this operation (*tabu*2), which contains a list of {teacher *i*, course *j*, section *k*, day *l*, and starting time period *m*} visited in the last *length*2 iterations. For example, if Course1 Section1 taught by Teacher1 has just been moved from Period1 to Period2, one could declare that moving back that particular course from Period2 to Period1 is tabu for a certain number of iterations.

Similar to Algorithm 2, if the first operation is rejected, the second operation is still performed by choosing section *k* of course *j* randomly, where $k \in K_j$. This course section will then be allocated to other time periods. The operation is continued by evaluating the objective function value and checking the tabu list, *tabu*3. In this operation, the tabu list contains a list of {course *j*, day *l*, and starting time period *m*}, which is forbidden in the last *length*3 iterations.

Figures 5.8 and 5.9 represent the pseudocode of Algorithms SA-TS (Algorithm 5) and the evaluation process (Algorithm 6), respectively. The details of both algorithms are presented as flow charts in Appendix *B*5 and *B*6.

Algorithm 5:

SA-TS ( )

(1) Initialize the parameters
(2) Set the best solution, *best_sol* = *initial_sol*
(3) Set the current solution, *current_sol* = *initial_sol*
(4) Set the total number of iterations, *num_iter* = 0
(5) Set the number of non improvement iterations, *no_improv* = 0
(6) **While** the total number of iterations, *num_iter* is less than the preset maximum number of iterations, *outer_loop* **do**:
(7) **Repeat** *inner_loop* times:
(8) Apply the first operation
(9) **If** the first operation is successful
(10) Evaluate the change by applying **Algorithm 6**
(11) Apply the second operation
(12) **If** the second operation is successful
(13) Evaluate the change by applying **Algorithm 6**
(14) **Else**
(15) Return to Step (8)
(16) Update temperature T_{num_iter}
(17) **If** *current_sol* is worse than *best_sol*
(18) *no_improv* := *no_improv* + 1
(19) **If** *no_improv* > the maximum number of non improvement iterations, *limit*
(20) Apply the intensification strategy
(21) *num_iter* := *num_iter* + 1
(22) **Endwhile**
(23) Report the best solution, *best_sol*

Figure 5.8 Pseudocode of Algorithm SA-TS

Algorithm 6:

EVALUATION PROCESS SA-TS ( )

(1) Calculate the new solution, *new_sol*
(2) Calculate the change of the objective function, Δ := *new_sol* – *current_sol*
(3) **If** $\Delta > 0$
(4) Update the current solution, *current_sol*
(5) **If** *current_sol* is better than *best_sol*
(6) Update the best solution, *best _sol* = *current_sol*
(7) Update tabu list
(8) **Else**
(9) Choose a random number r_1 uniformly from [0,1]
(10) Check whether the new solution is tabu
(11) **If** $r_1 < exp^{\Delta/T_{num\_iter}}$ and the new solution is not tabu
(12) Accept the new solution, *new_sol*
(13) Update the current solution, *current_sol*
(14) Update tabu list
(15) **Else**
(16) Return to the current solution, *current_sol*
(17) Update tabu list

Figure 5.9 Evaluation process of Algorithm SA-TS

5.3 Computational Results

The computational experiments were performed on a 2.6GHz Intel Pentium IV PC with 512 MB RAM under the Microsoft Windows XP Operating System. The proposed algorithms were coded in C++. The solution of the TA model is obtained by ILOG OPL Studio 4.2 Solver.

Our tests are performed over the same data sets used in Chapter 4. The values of the parameters used by the proposed algorithms are summarized in Table 5.2. Preliminary experimentation was performed to determine suitable values for the parameters of the proposed heuristic. These values were chosen to ensure a compromise between the computation time and the solution quality. This single set of SA parameter settings is used for comparing all the proposed algorithms on the same data sets.

Table 5.2 Parameter settings for hybrid algorithms

Parameter	Value
Number of iterations, *outer_loop*	$\|I\|\times\|L\|\times\|M\|$
Initial temperature, T_0	10,000
Number of neighbor moves at each temperature T_n, *inner_loop*	$\|I\|\times\|L\|\times\|M\|$
Cooling factor, α	0.95
Number of non-improvement iterations prior to intensification, *limit*	$0.05\times\|I\|\times\|L\|\times\|M\|$
Length of *tabu*1, *length*1	$0.25\times\|I\|$ for Group I data sets, $0.5\times\|I\|$ for Group II data sets
Length of *tabu*2, *length*2	$\|L\|$ for Group I data sets, $2\times\|L\|$ for Group II data sets
Length of *tabu*3, *length*3	$\|L\|$ for Group I data sets, $2\times\|L\|$ for Group II data sets

To see if the proposed algorithms have better than simply solving the teacher assignment and course scheduling problems separately as was commonly seen in the literatures discussed in Chapter 2, we have performed computational experiments on solving the Group I data sets via

the latter approach. This involved solving the TA Model and then using the results obtained as the inputs for the TACS Model III.

For each data set, the proposed algorithms were executed 20 times with different random seeds. Tables 5.3 summarizes the overall results that include the objective function values obtained by solving the teacher assignment and course scheduling sub-problems separately and by solving the TACS Model III, the average CPU time required to solve the TACS Model III, as well as the average objective function value obtained, the best objective function value obtained and the average CPU time required to obtain the solution by Algorithms SA1, SA2 and SA-TS. Table 5.4 represents similar results for Group II data sets.

From Table 5.3, it can be seen that solving the teacher assignment and course scheduling sub-problems separately is inferior in performance than the proposed algorithms in terms of the objective function values obtained for all the Group I data sets.

We observe that the computation time taken by Algorithm SA1 to obtain the solution is less than the computation time taken by Algorithm SA2 and Algorithm SA-TS for all the data sets. It is due to the evaluation process is run twice in the improvement phase of Algorithms SA2 and SA-TS. However, better results have been obtained by Algorithm SA2 and Algorithm SA-TS since both algorithms are able to obtain the optimal solution for most of group I data sets.

Table 5.3 Computational results of Algorithms SA1, SA2 and SA-TS on Group I data sets

Data Set	Solution obtained by commercial software			Algorithm SA1			Algorithm SA2			Algorithm SA-TS		
	Objective function value for separating model solving	Objective function value for solving TACS′ Model III	CPU time (seconds)	Average objective function value	Best objective function value	Average CPU time (seconds)	Average objective function value	Best objective function value	Average CPU time (seconds)	Average objective function value	Best objective function value	Average CPU time (seconds)
5×5_1(1)	1100	1130	2.43	1100	1100	0.09	1107.5	1130	0.13	1100	1100	0.75
5×5_1(2)	900	1090	2.23	910	910	0.10	1086.5	1090	0.14	1088.5	1090	0.78
5×5_1(3)	880	1040	1.65	880	880	0.08	1020	1020	0.13	1040	1040	0.76
5×5_1(4)	800	1020	2.04	800	800	0.06	1011.5	1020	0.13	1012	1020	0.77
5×5_1(5)	930	980	1.82	930	930	0.09	950	950	0.14	980	980	0.77
5×5_2(1)	1150	1280	1.85	1170	1170	0.07	1230	1230	0.12	1225.5	1230	0.68
5×5_2(2)	1110	1290	2.45	1125.5	1180	0.09	1290	1290	0.12	1290	1290	0.72
5×5_2(3)	1000	1110	2.25	994	1040	0.08	1108	1110	0.12	1097	1110	0.73
5×5_2(4)	820	1190	2.03	810	810	0.05	1100	1100	0.12	1148.5	1160	0.66
5×5_2(5)	1130	1210	2.25	1130	1130	0.08	1160	1160	0.12	1210	1210	0.71
10×10_1(1)	2230	2340	31.07	2230	2230	1.58	2290	2290	4.94	2325	2340	5.12
10×10_1(2)	1800	2260	11.59	1820.5	1830	1.56	2236.5	2260	5.10	2242.5	2260	5.44
10×10_1(3)	2020	2200	22.15	2020	2020	1.59	2192	2200	5.24	2200	2200	5.91
10×10_1(4)	1790	2020	12.50	1780	1780	1.57	2020	2020	5.45	2019	2020	5.57
10×10_1(5)	1850	2100	15.03	1880	1880	1.55	2025.5	2090	5.29	2090	2090	5.65
10×10_2(1)	2650	2860	49.97	2700	2700	1.47	2842.5	2850	4.70	2845.5	2850	4.98
10×10_2(2)	2340	2780	32.29	2340	2340	1.50	2725.5	2780	4.47	2764	2780	4.51
10×10_2(3)	2560	2780	19.70	2560	2560	1.48	2766.5	2780	4.39	2777.5	2780	4.39
10×10_2(4)	2250	2540	10.96	2260	2260	1.49	2468.5	2500	4.75	2478.5	2500	5.01
10×10_2(5)	2300	2600	46.03	2303.5	2370	1.47	2547	2550	4.78	2547	2550	5.14

Table 5.3 Computational results of Algorithms SA1, SA2 and SA-TS on Group I data sets (*continued*)

Data Set	Solution obtained by commercial software			Algorithm SA1			Algorithm SA2			Algorithm SA-TS		
	Objective function value for separating model solving	Objective function value for solving TACS′ Model III	CPU time (seconds)	Average objective function value	Best objective function value	Average CPU time (seconds)	Average objective function value	Best objective function value	Average CPU time (seconds)	Average objective function value	Best objective function value	Average CPU time (seconds)
15×15_1(1)	2900	3080	67.23	2950	2950	5.25	3062.5	3070	17.40	3069	3070	19.65
15×15_1(2)	3140	3270	200.32	3140	3140	2.30	3267.5	3270	16.37	3270	3270	18.18
15×15_1(3)	2810	3130	222.63	2830	2830	5.16	3120	3120	17.51	3119	3130	19.84
15×15_1(4)	2970	3280	70.53	3020	3020	5.21	3196.5	3200	18.09	3212.5	3270	21.75
15×15_1(5)	2740	3090	123.95	2740	2740	5.20	3030	3050	19.85	3039	3080	19.57
15×15_2(1)	3790	4170	265.11	3920	3920	4.93	4068	4110	15.77	4099.5	4120	18.38
15×15_2(2)	3860	4260	554.36	3930	3930	4.93	4177.5	4220	15.43	4223	4230	18.43
15×15_2(3)	3850	4150	189.92	3850	3850	4.91	4140	4150	18.12	4141	4150	18.78
15×15_2(4)	4010	4320	401.42	4028	4090	4.79	4241	4310	15.60	4280.5	4300	17.78
15×15_2(5)	3510	3930	748.90	3553	3560	6.62	3780.5	3850	15.41	3844.5	3900	17.59
20×20_1(1)	4140	4290	583.82	4220	4220	11.08	4289.5	4290	29.78	4289	4290	31.23
20×20_1(2)	3560	4330	177.10	3760	3760	10.93	4213	4220	30.24	4220	4250	31.67
20×20_1(3)	4070	4340	444.72	3920	3920	10.90	4303	4320	31.25	4294.5	4320	31.39
20×20_1(4)	4420	4540	172.26	4420	4420	10.94	4540	4540	31.18	4540	4540	30.63
20×20_1(5)	4270	4340	312.01	4070	4070	10.99	4225	4280	29.20	4230	4300	30.38
20×20_2(1)	5460	5660	6637.76	5460	5460	10.98	5512.5	5570	28.25	5574.5	5610	30.30
20×20_2(2)	5500	5730	2127.28	4990	4990	10.19	5505	5570	27.98	5554.5	5630	28.58
20×20_2(3)	5650	5770	9367.78	5130	5130	10.16	5483	5600	26.84	5482.5	5600	28.30
20×20_2(4)	5540	5880	379.04	5546	5550	10.39	5699.5	5760	26.85	5798	5850	29.68
20×20_2(5)	5360	5680	2552.72	5289	5310	10.27	5556	5620	27.05	5560.5	5620	28.18

Table 5.4 Computational results of Algorithms SA1, SA2 and SA-TS on Group II data sets

Data Set	Solution obtained by commercial software		Algorithm SA1			Algorithm SA2			Algorithm SA-TS		
	Objective function value for solving TACS Model III	CPU time (seconds)	Average objective function value	Best objective function value	Average CPU time (seconds)	Average objective function value	Best objective function value	Average CPU time (seconds)	Average objective function value	Best objective function value	Average CPU time (seconds)
10×20_1(1)	7590	1578.91	6537.5	6650	4.04	7162.5	7240	6.42	7145.5	7250	6.55
10×20_1(2)	8210	588.28	6904.5	6960	2.41	7493.5	7600	5.70	7563	7720	5.56
10×20_1(3)	7670	311.59	6471.5	6500	2.43	6959	7040	5.88	7108.5	7200	5.86
10×20_1(4)	7560	2789.25	6312	6410	4.04	6810.5	6950	5.95	6907.5	7070	6.24
10×20_1(5)	7800	647.54	6913	6940	3.79	7340.5	7410	6.09	7429.5	7490	6.47
10×20_2(1)	7660	11145.60	6598	6670	3.98	7250	7330	6.38	7186	7360	6.81
10×20_2(2)	8480	423.92	7081.5	7110	2.17	7522.5	7630	5.70	7655.5	7760	5.50
10×20_2(3)	7970	541.53	6270.5	6480	2.18	6906.5	7060	5.91	7033	7160	5.95
10×20_2(4)	7730	3473.21	6434	6620	4.23	6941.5	7050	5.98	7113	7210	6.29
10×20_2(5)	7830	812.51	6467.5	6790	3.60	7136.5	7250	6.03	7245	7320	6.65
20×30_1(1)	11010	10283.73	9774	9800	6.74	10512.5	10720	19.56	10568.5	10680	35.26
20×30_1(2)	11140	9265.63	9621	9680	6.72	10597.5	10760	19.41	10718.5	10920	37.73
20×30_1(3)	10430	10598.13	9300	9300	6.69	9779	9870	17.54	9812	9930	30.59
20×30_1(4)	10920	18584.29	9333.5	9400	6.96	9868	10040	18.15	10085	10230	32.44
20×30_1(5)	11260	9469.55	9787.5	9810	6.73	10569	10740	18.56	10711	10880	33.34
20×30_2(1)	13090	17184.70	11456.5	11770	10.76	12298.5	12460	26.91	12424	12580	66.36
20×30_2(2)	12880	55032.51	11132	11370	10.96	12206	12460	27.44	12254.5	12390	29.79
20×30_2(3)	13660	47060.92	11439	11460	12.82	12231.5	12490	26.67	12287.5	12490	48.03
20×30_2(4)	14260	38239.02	11822	11860	11.07	12997.5	13120	27.25	13155	13250	49.05
20×30_2(5)	13450	22582.79	10967	11440	10.65	11990	12200	25.24	12165.5	12410	49.05

Table 5.4 Computational results of Algorithms SA1, SA2 and SA-TS on Group II data sets (*continued*)

Data Set	Solution obtained by commercial software		Algorithm SA1			Algorithm SA2			Algorithm SA-TS		
	Objective function value for solving TACS′ Model III	CPU time (seconds)	Average objective function value	Best objective function value	Average CPU time (seconds)	Average objective function value	Best objective function value	Average CPU time (seconds)	Average objective function value	Best objective function value	Average CPU time (seconds)
20×40_1(1)	13210[a]	-[b]	11680.5	11940	13.45	12568.5	12760	73.35	12704	12920	128.36
20×40_1(2)	14010[a]	-[b]	11671.5	11880	10.88	12484.5	12760	83.86	12654.5	12890	139.92
20×40_1(3)	13360[a]	-[b]	11821.5	11850	11.92	12427.5	12550	78.11	12547	12720	132.54
20×40_1(4)	13860[a]	-[b]	11939.5	12010	10.69	12818.5	12900	78.15	12844	12990	138.97
20×40_1(5)	14270[a]	-[b]	12088	12210	10.98	12979	13240	80.26	12995	13220	137.97
20×40_2(1)	16190[a]	-[b]	14136	14200	19.86	14683	14800	46.18	14957.5	15120	101.42
20×40_2(2)	17440[a]	-[b]	14130.5	14390	20.06	15283.5	15520	43.74	15462	15600	107.91
20×40_2(3)	17240[a]	-[b]	14388.5	14540	19.63	15170.5	15400	45.00	15423.5	15540	104.45
20×40_2(4)	16700[a]	-[b]	13720	13880	20.09	14625	14880	43.10	14856.5	15060	104.26
20×40_2(5)	16690[a]	-[b]	13336	14110	22.50	14711	14970	42.43	15034	15250	99.74
30×60_2(1)	20540[a]	-[b]	18224	18280	56.14	19253	19470	193.53	19346.5	19530	389.06
30×60_2(2)	21060[a]	-[b]	18328	18370	52.75	19472	19660	199.59	19695	19830	477.57
30×60_2(3)	20710[a]	-[b]	18463	18640	53.20	19161.5	19330	196.52	19375.5	19550	390.22
30×60_2(4)	21890[a]	-[b]	19250	19300	52.01	20486	20650	211.55	20806.5	20910	379.60
30×60_2(5)	21170[a]	-[b]	18336	18510	53.87	19460	19640	194.13	19749.5	20010	359.04
30×60_1(1)	-	-[b]	21400	21880	83.37	23218	23780	258.07	23498.5	23940	387.13
30×60_1(2)	25180[a]	-[b]	20645	21050	85.02	22480.5	22870	245.35	22782.5	23200	487.88
30×60_1(3)	-	-[b]	21034	21330	78.24	22613.5	22840	255.08	22959	23270	418.27
30×60_1(4)	-	-[b]	21820	21820	79.16	23414.5	23600	243.30	23583.5	23840	384.61
30×60_1(5)	24480[a]	-[b]	20643	20930	78.98	22428.5	23060	241.03	22793.5	23180	394.47

[a] *The best known solution obtained within 24 hours*
[b] *CPU time = 24 hours*

A comparison of the objective function values and computation times (in seconds) obtained by the proposed algorithms and the best known/optimal solutions was summarized in Tables 5.5 and 5.6. We use the same formulae presented in Chapter 4 to calculate the deviations of the best and the average objective function values of the proposed algorithm from the best known/optimal value.

Table 5.5 Comparison of the algorithm results and the optimal solutions on Group I data sets

Data Set	Algorithm SA1		Algorithm SA2		Algorithm SA-TS	
	$\Phi_{(1)}^{SA1}$ (%)	$\Phi_{(2)}^{SA1}$ (%)	$\Phi_{(1)}^{SA2}$ (%)	$\Phi_{(2)}^{SA2}$ (%)	$\Phi_{(1)}^{SA-TS}$ (%)	$\Phi_{(2)}^{SA-TS}$ (%)
5×5_1(1)	2.65	2.65	1.99	0.00	2.65	2.65
5×5_1(2)	16.51	16.51	0.32	0.00	0.14	0.00
5×5_1(3)	15.38	15.38	1.92	1.92	0.00	0.00
5×5_1(4)	21.57	21.57	0.83	0.00	0.78	0.00
5×5_1(5)	5.10	5.10	3.06	3.06	0.00	0.00
5×5_2(1)	8.59	8.59	3.91	3.91	4.26	3.91
5×5_2(2)	12.75	8.53	0.00	0.00	0.00	0.00
5×5_2(3)	10.45	6.31	0.18	0.00	1.17	0.00
5×5_2(4)	31.93	31.93	7.56	7.56	3.49	2.52
5×5_2(5)	6.61	6.61	4.13	4.13	0.00	0.00
10×10_1(1)	4.70	4.70	2.14	2.14	0.64	0.00
10×10_1(2)	19.45	19.03	1.04	0.00	0.77	0.00
10×10_1(3)	8.18	8.18	0.36	0.00	0.00	0.00
10×10_1(4)	11.88	11.88	0.00	0.00	0.05	0.00
10×10_1(5)	10.48	10.48	3.55	0.48	0.48	0.48
10×10_2(1)	5.59	5.59	0.61	0.35	0.51	0.35
10×10_2(2)	15.83	15.83	1.96	0.00	0.58	0.00
10×10_2(3)	7.91	7.91	0.49	0.00	0.09	0.00
10×10_2(4)	11.02	11.02	2.81	1.57	2.42	1.57
10×10_2(5)	11.40	8.85	2.04	1.92	2.04	1.92
15×15_1(1)	4.22	4.22	0.57	0.32	0.36	0.32
15×15_1(2)	3.98	3.98	0.08	0.00	0.00	0.00
15×15_1(3)	9.58	9.58	0.32	0.32	0.35	0.00
15×15_1(4)	7.93	7.93	2.55	2.44	2.06	0.30
15×15_1(5)	11.33	11.33	1.94	1.29	1.65	0.32
15×15_2(1)	6.00	6.00	2.45	1.44	1.69	1.20
15×15_2(2)	7.75	7.75	1.94	0.94	0.87	0.70
15×15_2(3)	7.23	7.23	0.24	0.00	0.22	0.00
15×15_2(4)	6.76	5.32	1.83	0.23	0.91	0.46
15×15_2(5)	9.59	9.41	3.80	2.04	2.18	0.76
20×20_1(1)	1.63	1.63	0.01	0.00	0.02	0.00
20×20_1(2)	13.16	13.16	2.70	2.54	2.54	1.85
20×20_1(3)	9.68	9.68	0.85	0.46	1.05	0.46
20×20_1(4)	2.64	2.64	0.00	0.00	0.00	0.00
20×20_1(5)	6.22	6.22	2.65	1.38	2.53	0.92
20×20_2(1)	3.53	3.53	2.61	1.59	1.51	0.88
20×20_2(2)	12.91	12.91	3.93	2.79	3.06	1.75
20×20_2(3)	11.09	11.09	4.97	2.95	4.98	2.95
20×20_2(4)	5.68	5.61	3.07	2.04	1.39	0.51
20×20_2(5)	6.88	6.51	2.18	1.06	2.10	1.06
Average	9.64	9.31	1.94	1.27	1.24	0.70
Maximum	31.93	31.93	7.56	7.56	4.98	3.91
Minimum	1.63	1.63	0.00	0.00	0.00	0.00

Table 5.6 Comparison of the algorithm results and the optimal solutions on Group II data sets

Data Set	Algorithm SA1		Algorithm SA2		Algorithm SA-TS	
	$\Phi_{(1)}^{SA1}$ (%)	$\Phi_{(2)}^{SA1}$ (%)	$\Phi_{(1)}^{SA2}$ (%)	$\Phi_{(2)}^{SA2}$ (%)	$\Phi_{(1)}^{SA-TS}$ (%)	$\Phi_{(2)}^{SA-TS}$ (%)
10×20_1(1)	13.87	12.38	5.63	4.61	5.86	4.48
10×20_1(2)	15.90	15.23	8.73	7.43	7.88	5.97
10×20_1(3)	15.63	15.25	9.27	8.21	7.32	6.13
10×20_1(4)	16.51	15.21	9.91	8.07	8.63	6.48
10×20_1(5)	11.37	11.03	5.89	5.00	4.75	3.97
10×20_2(1)	13.86	12.92	5.35	4.31	6.19	3.92
10×20_2(2)	16.49	16.16	11.29	10.02	9.72	8.49
10×20_2(3)	21.32	18.70	13.34	11.42	11.76	10.16
10×20_2(4)	16.77	14.36	10.20	8.80	7.98	6.73
10×20_2(5)	17.40	13.28	8.86	7.41	7.47	6.51
20×30_1(1)	11.23	10.99	4.52	2.63	4.01	3.00
20×30_1(2)	13.64	13.11	4.87	3.41	3.78	1.97
20×30_1(3)	10.83	10.83	6.24	5.37	5.93	4.79
20×30_1(4)	14.53	13.92	9.63	8.06	7.65	6.32
20×30_1(5)	13.08	12.88	6.14	4.62	4.88	3.37
20×30_2(1)	12.48	10.08	6.05	4.81	5.09	3.90
20×30_2(2)	13.57	11.72	5.23	3.26	4.86	3.80
20×30_2(3)	16.26	16.11	10.46	8.57	10.05	8.57
20×30_2(4)	17.10	16.83	8.85	7.99	7.75	7.08
20×30_2(5)	18.46	14.94	10.86	9.29	9.55	7.73
20×40_1(1)	11.58	9.61	4.86	3.41	3.83	2.20
20×40_1(2)	16.69	15.20	10.89	8.92	9.68	7.99
20×40_1(3)	11.52	11.30	6.98	6.06	6.09	4.79
20×40_1(4)	13.86	13.35	7.51	6.93	7.33	6.28
20×40_1(5)	15.29	14.44	9.05	7.22	8.93	7.36
20×40_2(1)	12.69	12.29	9.31	8.59	7.61	6.61
20×40_2(2)	18.98	17.49	12.37	11.01	11.34	10.55
20×40_2(3)	16.54	15.66	12.00	10.67	10.54	9.86
20×40_2(4)	17.84	16.89	12.43	10.90	11.04	9.82
20×40_2(5)	20.10	15.46	11.86	10.31	9.92	8.63
30×60_2(1)	11.28	11.00	6.27	5.21	5.81	4.92
30×60_2(2)	12.97	12.77	7.54	6.65	6.48	5.84
30×60_2(3)	10.85	10.00	7.48	6.66	6.44	5.60
30×60_2(4)	12.06	11.83	6.41	5.66	4.95	4.48
30×60_2(5)	13.39	12.56	8.08	7.23	6.71	5.48
30×60_1(1)	-	-	-	-	-	-
30×60_1(2)	18.01	16.40	10.72	9.17	9.52	7.86
30×60_1(3)	-	-	-	-	-	-
30×60_1(4)	-	-	-	-	-	-
30×60_1(5)	15.67	14.50	8.38	5.80	6.89	5.31
Average	14.85	13.69	8.47	7.13	7.41	6.13
Maximum	21.32	18.70	13.34	11.42	11.76	10.55
Minimum	10.83	9.61	4.52	2.63	3.78	1.97

The tables above show differences of the quality of solutions produced by each algorithm. SA1 does not perform very well with the maximum values of both parameters $\Phi_{(1)}^{SA1}$ and $\Phi_{(2)}^{SA1}$ are 31.93%. A possible reason is that the exploration of the neighborhood moves in SA1 is limited. The neighborhood consists of two operations: reallocation of teachers and time periods. If the first operation is accepted, we continue to reallocate the course to other time

periods. Unfortunately, it might be possible that the teacher reallocation is rejected especially during the iterations with low temperature T and the process is then terminated without proceeding to the second operation.

The experiments clearly demonstrate the superiority of Algorithm SA2 and SA-TS. They yield the optimal solutions for most of Group I data sets. Similar to Group I results, SA1 could not perform very well with the minimum values of $\Phi_{(1)}^{SA1}$ and $\Phi_{(2)}^{SA1}$ are 10.83% and 9.61%, respectively. The results obtained by SA2 and SA–TS show some improvement over those of SA1. The best performance is shown by Algorithm SA–TS where the best value of $\Phi_{(2)}^{SA-TS}$ is only 1.97%. Some features of Tabu Search incorporated in Algorithm SA-TS are able to prevent unnecessary moves during high temperature and lead to better solutions.

In summary, we observe that solution quality depends not only on the evaluation process but also on the way in which it is applied. Overall, these experiments indicate that the results of SA2 and SA-TS are vastly superior to those obtained by SA1. However, this improvement is obtained at the cost of additional computation time.

5.4 Conclusions

This chapter presented three different hybrid algorithms, known as SA1, SA2 and SA-TS, to solve the TACS problem. They combine a greedy heuristic, Simulated Annealing and Tabu Search algorithms.

The entire problem was divided into two interrelated sub-problems, teacher assignment and course scheduling problems. The teacher assignment problem was solved by ILOG OPL Studio software and the course scheduling problem was tackled by a greedy heuristic.

The main difference of the pure SA and our proposed algorithms lies in the additional strategy applied. The intensification strategy of the Tabu Search algorithm is included in each proposed algorithm, Algorithms SA1, SA2 and SA-TS. The idea of the intensification strategy is to focus the search once again starting from the best solution obtained in order to further improve the quality of the solutions if there is no improvement of the solution obtained after a certain number of iterations.

There are two operations involved in each algorithm: reallocation of teachers to courses and course sections and reallocation course sections to time periods. In Algorithm SA1, the acceptance – rejection process (the evaluation process) of SA is only applied at the end of both operations. In Algorithms SA2 and SA-TS, the evaluation process is applied at the end of each operation. We conclude that by conducting the evaluation process at the end of each operation, better solutions would be obtained. Algorithms SA2 and SA-TS are able to explore more possible neighborhood moves.

Some features of Tabu Search, such as aspiration criterion, tabu length and tabu list is included in the evaluation process of Algorithm SA-TS. It has been shown that Algorithms SA-TS outperforms SA1 and SA2 in terms of objective function values obtained. Tabu list is able to avoid excessive or unnecessary moves especially during high temperatures, which will deteriorate the objective function value.

Another hybridization of Lagrangian relaxation and metaheuristics that focus on building the initial feasible solutions would be proposed in the next chapter. The performance would be compared with the one of the proposed algorithms, SA-TS.

CHAPTER VI
HYBRIDIZATION OF LAGRANGIAN RELAXATION AND METAHEURISTICS FOR THE TACS PROBLEM

6.1 Introduction

The main idea of the proposed algorithms explained in Chapter 5 is to focus on improving the initial solutions generated from the construction phase. The initial solution was initially generated by a greedy heuristic. In this Chapter, we focus on how to construct good quality initial solutions that would lead to better quality final solutions. The quality of the initial solution does play an important role in order to find better solutions (Joubert and Claasen, 2006; Moody et al., 2008).

In order to construct the initial solution, we apply the Lagrangian Relaxation approach (Held and Karp, 1970). The approach essentially consists of removing certain "complicating" constraints and incorporating them in the objective function so that the relaxed problem can be solved efficiently. The objective function value obtained is then treated as an upper bound on the optimal value of the original maximization problem.

A detailed review and successful applications of the Lagrangian relaxation have been documented by Fisher (1981). Several other successful applications of the Lagrangian relaxation approach have been reported by Geoffrion (1974), Fisher (1981), Chien et al. (1989), Brännlund et al. (1998) and Kim and Kim (2000).

In this study, we develop a hybridization of Lagrangian relaxation approach and Algorithm SA2, namely, Algorithm LR-SA, for solving TACS Model III. In the construction phase, the problem is solved by the Lagrangian relaxation approach for generating initial feasible solutions. These initial solutions are further improved by Algorithm SA2 in the improvement phase.

This chapter is organized as follows. Section 6.2 describes the entire proposed algorithm for solving the problem. In Section 6.3, an extensive computational evaluation of the proposed algorithm, including comparison and analysis of results obtained, is presented to illustrate the performance of the new approach. At the end of this chapter, some related discussions are summarized.

6.2 The Proposed Algorithm

Algorithm LR-SA consists of three phases: (1) pre-processing, (2) construction and (3) improvement. Each of these phases would be further described below.

6.2.1 The Pre-processing Phase

Similar to previous proposed algorithms (Algorithms SA1, SA2 and SA-TS), two different sets I_j and LM_i are generated.

6.2.2 The Construction Phase

The second phase of the hybrid algorithm is used primarily to obtain an initial feasible solution. The consecutiveness requirement is identified as a major source of complexity in TACS Model

III. This requirement turns the problem from easy to hard, i.e. from polynomially solvable to NP-hard (ten Eikelder and Willemen, 2001 and Daskalaki and Birbas, 2005). We decide to remove the constraints related to this requirement from the model and only consider them in the process of generating an initial feasible solution. Thus, we proposed another mathematical model, denoted as TACS Model IV, which is similar to TACS Model III without consecutiveness requirements (equations (3.37), (3.38), (3.40), (3.41) and (3.44)).

[TACS Model IV]
Maximize *Objective function* (3.22)
subject to:
Constraints (3.23) – (3.36), (3.39), (3.42) – (3.43), (3.45) – (3.47)

The TACS Model IV is further decomposed into two smaller models: TA Model and CS Model whereas each model represents the teacher assignment and course scheduling problems, respectively. These two models would be solved in a three-stage relaxation procedure. The first stage focuses on solving the TA Model. The results obtained are further used as an input in the second stage which is related to the CS Model.

We propose a Lagrangian relaxation approach in order to solve the CS Model. Since the results obtained might not be feasible to TACS Model III, we propose an additional heuristic, so called as a Lagrangian heuristic in the third stage. In this stage, we include the consecutiveness constraints in order to construct a feasible solution. The details of each stage would be described below.

6.2.2.1 The Teacher Assignment Sub-Problem

Here, we use the same model presented in Section 5.2.2 in order to obtain the solution of the teacher assignment problem, X'_{ijk}. TA Model is then optimally solved by CPLEX 10.0 and the decision variable X'_{ijk} is used as an input in the next model, CS Model.

6.2.2.2 The Course Scheduling Sub-Problem

We propose another mathematical model by considering the requirements of the course scheduling problem, denoted as CS Model. As mentioned earlier, the consecutiveness requirements has been removed and would only be considered in the process of generating an initial feasible solution based on a Lagrangian heuristic.

[CS Model]

Maximize $$Z_{CS} = \sum_{i\in I}\sum_{j\in J}\sum_{k\in K_j}\sum_{l\in L}\sum_{m\in M} PT^{\mathrm{III}}_{ilm} \times X_{ijklm} \tag{6.1}$$

subject to:

$$\sum_{l\in L} Y_{ijkl} = X'_{ijk} \qquad \left(i \in I, \forall j \in J, \forall k \in K_j\right) \tag{6.2}$$

$$\sum_{i\in I}\sum_{k\in K_j} Y_{ijkl} \le 1 \qquad \left(j \in J, \forall l \in L\right) \tag{6.3}$$

$$\sum_{i\in I}\sum_{k\in K_j}\sum_{l\in L} Y_{ijkl} = Sec_j \qquad \left(j \in J\right) \tag{6.4}$$

$$\sum_{i\in I}\sum_{l\in L} Y_{ijkl} = 1 \qquad \left(j \in J, \forall k \in K_j\right) \tag{6.5}$$

$$\sum_{j\in J}\sum_{k\in K_j}\sum_{l\in L} Y_{ijkl} = L_i \qquad \left(i \in I\right) \tag{6.6}$$

$$V_i = \left\lceil \frac{L_i}{5} \right\rceil \qquad \left(i \in I\right) \tag{6.7}$$

$$\sum_{j\in J}\sum_{k\in K_j} Y_{ijkl} \le V_i \qquad \left(i \in I, \forall l \in L\right) \tag{6.8}$$

$$\sum_{m\in M} X_{ijklm} = Y_{ijkl} \times H_j \qquad \left(i \in I, \forall j \in J, \forall k \in K_j, \forall l \in L\right) \tag{6.9}$$

$$\sum_{i\in I}\sum_{k\in K_j} X_{ijklm} \le 1 \qquad \left(j \in J, \forall l \in L, \forall m \in M\right) \tag{6.10}$$

$$\sum_{j\in J}\sum_{k\in K_j} X_{ijklm} \le 1 \qquad \left(i \in I, \forall l \in L, \forall m \in M\right) \tag{6.11}$$

$$\sum_{i\in I}\sum_{j\in J}\sum_{k\in K} X_{ijklm} \le C_{lm}^{\mathrm{III}} \qquad \left(l \in L, \forall m \in M\right) \tag{6.12}$$

$$\sum_{i\in I}\sum_{l\in L}\sum_{m\in M} X_{ijklm} = H_j \qquad \left(j \in J, \forall k \in K_j\right) \tag{6.13}$$

$$Y_{ijkl} \in \{0,1\} \qquad \left(i \in I, \forall j \in J, \forall k \in K_j, \forall l \in L\right) \tag{6.14}$$

$$L_i \in Z^+ \qquad \left(i \in I\right) \tag{6.15}$$

$$V_i \in Z^+ \qquad \left(i \in I\right) \tag{6.16}$$

$$X_{ijklm} \in \{0,1\} \qquad \left(i \in I, \forall j \in J, \forall k \in K_j, \forall l \in L, \forall m \in M\right) \tag{6.17}$$

The objective function (6.1) only reflects the time preferences. An additional constraint (6.2) is introduced in order to relate the TA Model results obtained, X'_{ijk}, to this sub-problem. This constraint ensures that only course sections that have been allocated to teachers will be scheduled on a particular day. The rest of the constraints are similar to those of course scheduling constraints in TACS Model III.

CPLEX 10.0 could not solve CS Model optimally due to the complexity of the problem. Therefore, we proposed a Lagrangian relaxation approach in order to generate good upper bounds for the problem. The Lagrangian relaxation approach essentially relaxes the original problem by removing certain constraints and incorporating them in the objective function using Lagrangian multipliers. For any given set of multipliers, an upper bound on the CS Model is served by the relaxed problem's objective function.

Consider the Lagrangian relaxation that dualizes constraint (6.9) of CS Model using Lagrangian multipliers λ_{ijkl}. An important property in evaluating a relaxation is the amount of computation time required to obtain the solutions (Fisher, 1981; Chien et al., 1989). Equation (6.9) is considered as a difficult constraint. After incorporating that constraint in the objective function, the relaxed problem can be solved efficiently within a reasonable computation time. The relaxed problem, denoted as CS($\boldsymbol{\lambda}$) Model, is given below:

[CS($\boldsymbol{\lambda}$) Model]

$$\text{Maximize } Z_{CS}(\boldsymbol{\lambda}) = \sum_{i\in I}\sum_{j\in J}\sum_{k\in K_j}\sum_{l\in L}\sum_{m\in M} PT^{\text{III}}_{ilm} \times X_{ijklm} + \sum_{i\in I}\sum_{j\in J}\sum_{k\in K_j}\sum_{l\in L}\lambda_{ijkl} \times \left(Y_{ijkl} \times H_j - \sum_{m\in M} X_{ijklm} \right)$$

$$= \sum_{i\in I}\sum_{j\in J}\sum_{k\in K_j}\sum_{l\in L}\sum_{m\in M}\left(PT^{\text{III}}_{ilm} - \lambda_{ijkl}\right)\times X_{ijklm} + \sum_{i\in I}\sum_{j\in J}\sum_{k\in K_j}\sum_{l\in L}\lambda_{ijkl} \times \left(Y_{ijkl} \times H_j\right) \qquad (6.18)$$

subject to:

(6.2) – (6.8), (6.10) - (6.17)

The above mathematical model is decomposed into two independent sub-models: the first sub-model, CS1($\boldsymbol{\lambda}$) Model, is the model of creating daily schedules of course sections, while the second sub-model, CS2($\boldsymbol{\lambda}$) Model, is that of further scheduling course sections to time periods. For the given values of X'_{ijk} and λ_{ijkl}, the following sub-models are optimized:

[CS1($\boldsymbol{\lambda}$) Model]:

$$\text{Maximize } Z_{CS1}(\boldsymbol{\lambda}) = \sum_{i\in I}\sum_{j\in J}\sum_{k\in K_j}\sum_{l\in L}\lambda_{ijkl} \times \left(Y_{ijkl} \times H_j\right) \qquad (6.19)$$

subject to:

(6.2) – (6.8), (6.14) - (6.16)

[CS2($\boldsymbol{\lambda}$) Model]:

$$\text{Maximize } Z_{CS2}(\boldsymbol{\lambda}) = \sum_{i\in I}\sum_{j\in J}\sum_{k\in K_j}\sum_{l\in L}\sum_{m\in M}\left(PT^{\text{III}}_{ilm} - \lambda_{ijkl}\right)\times X_{ijklm} \qquad (6.20)$$

subject to:

(6.10) - (6.13), (6.17)

An upper bound on the optimal solution value of CS Model, $Z_{CS}(\lambda)$, can be computed by the following formula: $Z_{CS}(\lambda) = Z_{CS1}(\lambda) + Z_{CS2}(\lambda)$, where $Z_{CS1}(\lambda)$ and $Z_{CS2}(\lambda)$ represents the solution values of CS1(λ) Model and CS2(λ) Model, respectively. Finally, the upper bound for TACS Model IV as well as TACS Model III is calculated by $Z_{TA} + Z_{CS}(\lambda)$. The following figure represents the entire framework of the construction phase.

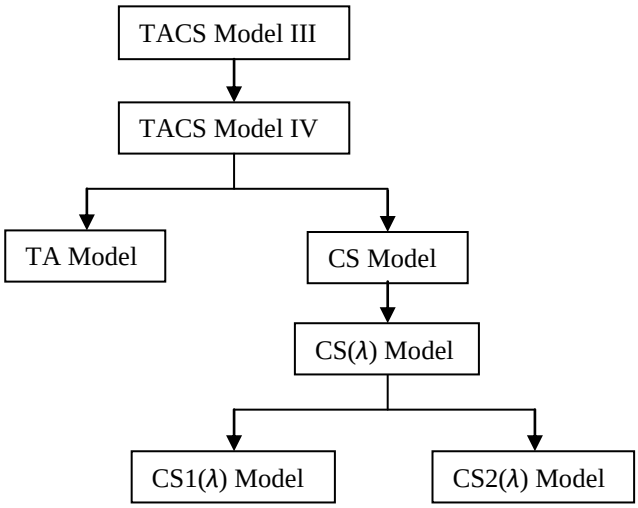


Figure 6.1 Framework of the construction phase

6.2.2.3 The Subgradient Search Method

The best objective function value can be obtained by finding good values for the Lagrangian multipliers, and the following dual problem [D] is used for this purpose.

$$[D] \quad \text{minimize}_{\lambda} \; Z_{CS}(\lambda) \tag{6.21}$$

In order to solve the dual problem [D], we use the sub-gradient search method as described in Fisher (1981). This method identifies good directions for changing the multipliers λ_{ijkl} and improving the upper bound. At each subgradient iteration, the best upper bound would be updated if the upper bound generated improves the current upper bound. The multipliers are updated with the following method:

$$\lambda_{ijkl} \leftarrow \lambda_{ijkl} + \theta \times \frac{\left(\sum_{m} X_{ijklm} - Y_{ijkl} \times H_j \right)}{\left\| DIFF \right\|^2} \qquad \left(i \in I, j \in J, k \in K_j, l \in L \right) \tag{6.22}$$

where,

- *DIFF* is the vector of differences between the left-hand sides and the right-hand sides of the constraint (6.9)
- λ_{ijkl} is initially set to zero

θ is the step size and is determined by $\theta = \delta \times \left(Z_{CS'}(\lambda) - Z_{LB} \right)$, where δ is a step-size multiplier such that $0 \le \delta \le 2$, and is Z_{LB} is the lower bound corresponding to the best known (heuristic) solution to original problem, CS Model. Initially, the value of δ is set to 2 (Fisher, 1981) and it would be adjusted by the following formula: $\delta \leftarrow 0.5 \times \delta$ whenever the objective function value of CS'(λ) Model has failed to improve in some specified number of iterations.

6.2.2.4 A Lagrangian Heuristic (Fisher, 1981)

The solutions of the relaxed problem CS(λ) Model may be infeasible to the original problem CS Model. As shown in Figure 6.1, since CS Model is part of TACS Model III, we develop a heuristic procedure, namely, Lagrangian heuristic, in order to convert an infeasible solution to a feasible solution for CS Model and TACS Model III as well.

In the Lagrangian heuristic, we include the consecutiveness constraints that have been removed in TACS Model IV. The best feasible solution of the Lagrangian heuristic is treated as the lower bound for CS Model. The lower bound of TACS Model III can be obtained by combining the lower bound for CS Model and the optimal solution of TA Model. This lower bound would then be treated as an initial feasible solution in the improvement phase.

In the Lagrangian heuristic, there are two processes involved. The first process examines whether the best solution of the relaxed problem CS(λ) is feasible for the original problem, TACS Model III, while the second process focuses on building a feasible solution. Each process would be described as follows.

The first process

Let decision variables Y_{ijkl} and X_{ijklm} be the best upper bound obtained so far in CS1(λ) and CS2(λ) Models. These decision variables might not be feasible for TACS Model III. In the first process, we define two different decision variables $\tilde{Y}_{ijkl}$ and $\tilde{X}_{ijklm}$ (with the default values are zero) that represent the final solutions of the Lagrangian heuristic. These variables would be further used as the initial feasible solution in the improvement phase.

We initially check whether each decision variable Y_{ijkl} (with value = 1) and the corresponding decision variables X_{ijklm} satisfy the relationship requirement and the consecutiveness requirement. For example, Let Y_{1111} = 1, we check whether $\sum_{m=1}^{|M|} X_{1111m} = Y_{1111} \times H_j$ (the relationship requirement). If it is satisfied, we continue to check whether $\sum_{m=m'}^{(m'+H_j-1)} X_{1111m} = H_j$ (the consecutiveness requirement). If these requirements are satisfied, we set the corresponding variables $\tilde{Y}_{ijkl}$ and $\tilde{X}_{ijklm}$ equal to 1.

At the end of the first process, we divide the decision variables $\tilde{Y}_{ijkl}$ into two different sets, defined by

$$S_1 = \left\{ (i,j,k) \middle| Y_{ijkl} = \tilde{Y}_{ijkl} = 1, i \in I, j \in J, k \in K_j, l \in L \right\} \quad (6.23)$$

$$S_2 = \left\{ (i,j,k) \middle| Y_{ijkl} = 1 \cap \tilde{Y}_{ijkl} = 0, i \in I, j \in J, k \in K_j, l \in L \right\} \quad (6.24)$$

Here, S_1 represents a set of course sections that have already been scheduled while S_2 represents a set of course sections that have not been scheduled yet. The details of the first process are presented in Figure 6.2 and Appendix *C*1.

***The first process**: (Check the feasibility of the solutions)*

(1) Define Y_{ijkl} and X_{ijklm} be the solution values of [CSP'1] and [CSP'2], respectively

(2) Define $\tilde{Y}_{ijkl}$ and $\tilde{X}_{ijklm}$ be the final solution values of [CSP]

(3) Define $S_1 = S_2 = \emptyset$

(4) Set $\tilde{Y}_{ijkl} = 0\left(i \in I, j \in J, k \in K_j, l \in L\right)$ and $\tilde{X}_{ijklm} = 0\left(i \in I, j \in J, k \in K_j, l \in L, m \in M\right)$

(5) **For** $\forall(i,j,k,l), i \in I, j \in J, k \in K_j, l \in L$ **do**

(6) **If** $Y_{ijkl} = 1$

(7) Calculate $\sum_{m \in M} X_{ijklm}$

(8) **If** $\sum_{m \in M} X_{ijklm} = Y_{ijkl} \times H_j$ where H_j is the number of time periods required for course *j* section *k*

(9) Find m', where m' is the first variable $X_{ijklm} = 1$

(10) Let $sum = 0$

(11) **For** $t = m'$ to $\left(m' + H_j - 1\right)$**do**

(12) **If** $X_{ijklt} = 1$

(13) $sum := sum + 1$

(14) **If** $sum = H_j$

(15) Set $\tilde{Y}_{ijkl} = 1$

(16) **For** $t = m'$ to $\left(m' + H_j - 1\right)$**do**

(17) Set $\tilde{X}_{ijklm} = 1$

(18) $S_1 = S_1 \cup (i,j,k)$

(19) **Else**

(20) $S_2 = S_2 \cup (i,j,k)$

(21) **Else**

(22) $S_2 = S_2 \cup (i,j,k)$

Figure 6.2 First process of a Lagrangian heuristic

The second process

The idea of the second process is to schedule a set of course sections in Set S_2. For each (i,j,k) in S_2, we perform a complete enumeration to find another possible set of day – time periods that satisfies the relationship and the consecutiveness constraints. However, we need to ensure that other requirements, such as the last period for conducting a course would not exceed the number of time period per day, capacity constraint and so forth, will not be violated. The details of the second process would be explained as follows.

The process is started by selecting one element (i,j,k) in S_2. This variable represents that teacher *i* teach course *j* section *k*. The process is continued by examining the first element (l_i, m_i)[1] from LM_i as the selected day and the starting time period. We examine whether that particular day l_i and time periods m_i to ($m_i + H_j - 1$) satisfy the following constraints:

- The length of time period required for course *j* section *k* will not exceed the number of time periods per day (Figure 6.3: Step 4).
- For course *j*, at most one section can be conducted per day (Figure 6.3: Step 10, corresponding to constraint (6.3)).
- The number of course sections taught by teacher *i* on day *l* has to be less than or equal to the maximum number of course sections taught per day, V_i (Figure 6.3: Step 16, corresponding to constraint (6.8)).
- For course *j*, at most one section can be taught at time period *m* (Figure 6.3: Step 22, corresponding to constraint (6.10)).
- Teacher *i* can only be assigned at most one course section on day *l* time period *m* (Figure 6.3: Step 27, corresponding to constraint (6.11)).
- The number of course sections taught on day *l* time period *m* cannot exceed the number of classroom available, *C* (Figure 6.3: Step 32, corresponding to constraint (6.12)).

If the first element (l_i, m_i)[1] satisfies the above-mentioned constraints, we set the corresponding variables $\tilde{Y}_{ijkl}$ and $\tilde{X}_{ijklm}$ equal to 1 and remove (i,j,k) from S_2. However, if the first element of LM_i is not feasible, the next element in LM_i with less preference values would be considered. Suppose that until the last element of LM_i, we still could not find any element (l_i, m_i), we need to relax some requirements and find the best possible allocation which refers to any element (l_i, m_i) that violates as few constraints as possible. Therefore, we impose some penalty values which related to infeasible solutions. This process is repeated until $S_2 = \emptyset$. The above method is similar to the process of allocating course sections to time periods presented in Section 4.2.2. The details of the second process are presented in Figure 6.3 and Appendix *C*2.

A Lagrangian heuristic is executed once after (N_H) iterations of sub-gradient search method. The entire construction phase is terminated if the number of iterations reaches a predetermined limit, N_K. Finally, the best feasible solution obtained by a Lagrangian heuristic is used as an initial feasible solution in the improvement phase. A Lagrangian heuristic has also been applied by Chien et al. (1989) for solving an integrated inventory allocation and vehicle routing problem and Kim and Kim (2000) for solving multi-period inventory and distribution planning problem.

***The second process*:** *(Build the feasible solution)*

(1) **For** $\forall(i,j,k)\in S_2$ **do**

(2) **Let** $t=1$

(3) Set d = t^{th} day component of LM_i and *start_time* = t^{th} time period component of LM_i

(4) Check whether $(start\_time + H_j - 1) \le |M|$

(5) **If** it is not satisfied

(6) **If** $t \le |L| \times |M|$, $t := t+1$, go to (3)

(7) **Else** go to (37)

(8) **Else if** it is satisfied

(9) For $u = start\_time$ to $(start\_time + H_j - 1)$

(10) Check whether $\sum_{o\in I}\sum_{p\in K_j}\tilde{Y}_{ojpd} \le 1$ (constraint (6.3) is satisfied)

(11) **If** it is not satisfied

(12) **If** $t \le |L| \times |M|$, $t := t+1$, go to (3)

(13) **Else** go to (37)

(14) **Else if** it is satisfied

(15) For $u = start\_time$ to $(start\_time + H_j - 1)$

(16) Check whether $\sum_{p\in J}\sum_{q\in K_{pj}}\tilde{Y}_{ipqd} + 1 \le V_i$ (constraint (6.8) is satisfied)

(17) **If** it is not satisfied

(18) **If** $t \le |L| \times |M|$, $t := t+1$, go to (3)

(19) **Else** go to (37)

(20) **Else if** it is satisfied

(21) For $u = start\_time$ to $(start\_time + H_j - 1)$

(22) Check whether $\sum_{o\in I}\sum_{q\in K_j}\tilde{X}_{ojkdu} \le 1$ (constraint (6.10) is satisfied)

(23) **If** it is not satisfied

(24) **If** $t \le |L| \times |M|$, $t := t+1$, go to (3)

(25) **Else** go to (37)

(26) **Else if** it is satisfied

(27) Check whether $\sum_{p\in J}\sum_{q\in K_p}\tilde{X}_{ipqdu} \le 1$ (constraint (6.11) is satisfied)

(28) **If** it is not satisfied

(29) **If** $t \le |L| \times |M|$, $t := t+1$, go to (3)

(30) **Else** go to (37)

(31) **Else if** it is satisfied

(32) Check whether $\sum_{o\in I}\sum_{p\in J}\sum_{q\in K_p}\tilde{X}_{opqdu} \le C$ (constraint (6.12) is satisfied)

(33) **If** it is not satisfied

(34) **If** $t \le |L| \times |M|$, $t := t+1$, go to (3)

(35) **Else** go to (37)

(36) **Else if** it is satisfied, go to (38)

(37) Relax some of the constraints violated and find the best possible allocation $(d, start\_time) \in LM^i$ where d and *start_time* represent the selected day and the starting time period, respectively, go to (38)

(38) Set $Y_{ijkd} = \tilde{Y}_{ijkd} = 1$ and $Y_{ijkl} = 0$

(39) **For** u = start_time to $(start\_time + H_j - 1)$

(40) Set $\tilde{X}_{ijklu} = 1$

(41) $S_1 = S_1 \cup (i,j,k)$ and $S_2 = S_2 \setminus (i,j,k)$

Figure 6.3 Second process of the Lagrangian heuristic

6.2.3 The Improvement Phase

At the end of previous phases, an initial feasible solution is obtained. This solution will be further improved in the improvement phase. We apply a modified Simulated Annealing (SA) algorithm which has attracted significant attention as an approach for large optimization problems (Ross, 2000). We apply the same idea of Algorithm SA2 proposed in Chapter 5. The details of Algorithm LR-SA are summarized in Appendix *C*3.

6.3 Computational Results

All the computational experiments were performed on a 2.6 GHz Intel Pentium IV PC with 512 MB of RAM running the Microsoft Windows XP Operating System. The proposed algorithm was coded in C++. CPLEX 10.0 is used for obtaining the solutions of the TA Model, CS1($\boldsymbol{\lambda}$) Model and CS2($\boldsymbol{\lambda}$) Model.

Computational experiments to evaluate the performance of the proposed algorithm were performed on two sets of different test problems presented in Chapter 3. The proposed algorithm requires few parameters to be set. Since we would like to make our results as reproducible as possible, we limit our runs in this experiment to a single set of parameter settings. Table 6.1 summarizes the values of the parameters used in the computational study which obtained throughout the experiment.

Table 6.1 Parameter setting for Algorithm LR-SA

Parameter	Value
Number of iterations needed before applying the heuristic procedure, N_H	20
Number of iterations in the construction phase, N_K	100
Number of iterations, *outer_loop*	$\|I\|\times\|L\|\times\|M\|$
Initial temperature, T_0	10,000
Number of neighbor moves at each temperature T_n, *inner_loop*	$\|I\|\times\|L\|\times\|M\|$
Cooling factor, α	0.95
Number of non-improvement iterations prior to intensification, *limit*	$0.05\times\|I\|\times\|L\|\times\|M\|$

The proposed algorithm was executed 20 times on each data set. Tables 6.2 and 6.3 summarize the overall results that include the objective function values obtained by solving the TACS′ Model, the average CPU time required to solve the TACS′ Model, as well as the average objective function value obtained, the best objective function value obtained and the average CPU time required to obtain the solution by both Algorithm SA-TS proposed in Chapter 5 and Algorithm LR-SA.

In most cases, the Algorithm LR-SA found better average objective values than those of Algorithm SA-TS except for the following data sets: 5×5_1 no 2, 10×10_2 no 3, 20×20_2 no 4, 30×60_2 no 1 and 4. In general, we conclude that Algorithm LR-SA outperforms Algorithm SA-TS.

Table 6.2 Computational results of Algorithm LR-SA on Group I data sets

Data Set	Solution obtained by commercial software		Algorithm SA-TS			Algorithm LR-SA		
	Objective function value	CPU time (seconds)	Average objective function value	Best objective function value	Average CPU time (seconds)	Average objective function value	Best objective function value	Average CPU time (seconds)
5×5_1(1)	1130	2.43	1100	1100	0.75	1130	1130	0.00
5×5_1(2)	1090	2.23	1088.5	1090	0.78	1065	1090	0.00
5×5_1(3)	1040	1.65	1040	1040	0.76	1040	1040	0.00
5×5_1(4)	1020	2.04	1012	1020	0.77	1020	1020	0.00
5×5_1(5)	980	1.82	980	980	0.77	980	980	0.00
5×5_2(1)	1280	1.85	1225.5	1230	0.68	1230	1230	0.00
5×5_2(2)	1290	2.45	1290	1290	0.72	1290	1290	0.00
5×5_2(3)	1110	2.25	1097	1110	0.73	1110	1110	0.00
5×5_2(4)	1190	2.03	1148.5	1160	0.66	1160	1160	0.00
5×5_2(5)	1210	2.25	1210	1210	0.71	1210	1210	0.00
10×10_1(1)	2340	31.07	2325	2340	5.12	2340	2340	6.7
10×10_1(2)	2260	11.59	2242.5	2260	5.44	2260	2260	5.65
10×10_1(3)	2200	22.15	2200	2200	5.91	2200	2200	5.4
10×10_1(4)	2020	12.50	2019	2020	5.57	2020	2020	7.55
10×10_1(5)	2100	15.03	2090	2090	5.65	2100	2100	6.55
10×10_2(1)	2860	49.97	2845.5	2850	4.98	2850	2850	7.4
10×10_2(2)	2780	32.29	2764	2780	4.51	2780	2780	6.25
10×10_2(3)	2780	19.70	2777.5	2780	4.39	2776.5	2780	3.65
10×10_2(4)	2540	10.96	2478.5	2500	5.01	2490	2490	5.6
10×10_2(5)	2600	46.03	2547	2550	5.14	2550	2550	10.35
15×15_1(1)	3080	67.23	3069	3070	19.65	3080	3080	41.8
15×15_1(2)	3270	200.32	3270	3270	18.18	3270	3270	41.1
15×15_1(3)	3130	222.63	3119	3130	19.84	3130	3130	31.8
15×15_1(4)	3280	70.53	3212.5	3270	21.75	3280	3280	39.3
15×15_1(5)	3090	123.95	3039	3080	19.57	3090	3090	43.5
15×15_2(1)	4170	265.11	4099.5	4120	18.38	4120	4120	42.4
15×15_2(2)	4260	554.36	4223	4230	18.43	4260	4260	39.75
15×15_2(3)	4150	189.92	4141	4150	18.78	4150	4150	32.35
15×15_2(4)	4320	401.42	4280.5	4300	17.78	4281.5	4300	33.55
15×15_2(5)	3930	748.90	3844.5	3900	17.59	3889	3890	29.4
20×20_1(1)	4290	583.82	4289	4290	31.23	4290	4290	67.6
20×20_1(2)	4330	177.10	4220	4250	31.67	4330	4330	74.85
20×20_1(3)	4340	444.72	4294.5	4320	31.39	4340	4340	67.3
20×20_1(4)	4540	172.26	4540	4540	30.63	4540	4540	68.8
20×20_1(5)	4340	312.01	4230	4300	30.38	4340	4340	75.5
20×20_2(1)	5660	6,637.76	5574.5	5610	30.30	5588	5620	77.9
20×20_2(2)	5730	2,127.28	5554.5	5630	28.58	5674	5690	69.4
20×20_2(3)	5770	9,367.78	5482.5	5600	28.30	5552	5620	68.4
20×20_2(4)	5880	379.04	5798	5850	29.68	5775	5830	61.3
20×20_2(5)	5680	2,552.72	5560.5	5620	28.18	5601	5630	63.1

Table 6.3 Computational results of Algorithm LR-SA on Group II data sets

Data Set	Solution obtained by commercial software		Algorithm SA-TS			Algorithm LR-SA		
	Objective function value	CPU time (seconds)	Average objective function value	Best objective function value	Average CPU time (seconds)	Average objective function value	Best objective function value	Average CPU time (seconds)
10×20_1(1)	7590	1578.91	7145.5	7250	6.55	7335	7350	12.08
10×20_1(2)	8210	588.28	7563	7720	5.56	7625	7740	17.34
10×20_1(3)	7670	311.59	7108.5	7200	5.86	7056	7290	15.9
10×20_1(4)	7560	2789.25	6907.5	7070	6.24	7105.5	7130	15.6
10×20_1(5)	7800	647.54	7429.5	7490	6.47	7591.5	7640	15.76
10×20_2(1)	7660	1114.60	7186	7360	6.81	7388.5	7420	14.04
10×20_2(2)	8480	423.92	7655.5	7760	5.50	7912	7950	15.48
10×20_2(3)	7970	541.53	7033	7160	5.95	7360.5	7420	13.18
10×20_2(4)	7730	3473.21	7113	7210	6.29	7310	7360	13.52
10×20_2(5)	7830	812.51	7245	7320	6.65	7358	7360	14.34
20×30_1(1)	11010	10283.73	10568.5	10680	35.26	10745.5	10770	91.95
20×30_1(2)	11140	9265.63	10718.5	10920	37.73	10940	10960	96.4
20×30_1(3)	10430	10598.13	9812	9930	30.59	10040	10040	82.75
20×30_1(4)	10920	18584.29	10085	10230	32.44	10432	10550	82.3
20×30_1(5)	11260	9469.55	10711	10880	33.34	10969.5	11090	82.05
20×30_2(1)	13090	17184.70	12424	12580	66.36	12454	12470	127.75
20×30_2(2)	12880	55032.51	12254.5	12390	29.79	12550	12550	121.9
20×30_2(3)	13660	47060.92	12287.5	12490	48.03	12904	13030	115.35
20×30_2(4)	14260	38239.02	13155	13250	49.05	13405	13410	125.2
20×30_2(5)	13450	22582.79	12165.5	12410	49.05	12539.5	12780	117.95
20×40_1(1)	13210[a]	-[b]	12704	12920	128.36	12927	13000	123.6
20×40_1(2)	14010[a]	-[b]	12654.5	12890	139.92	13256.5	13320	119.85
20×40_1(3)	13360[a]	-[b]	12547	12720	132.54	12648.5	12910	126.45
20×40_1(4)	13860[a]	-[b]	12844	12990	138.97	12875.5	13060	111.85
20×40_1(5)	14270[a]	-[b]	12995	13220	137.97	13646.5	13770	118.65
20×40_2(1)	16190[a]	-[b]	14957.5	15120	101.42	15484.5	15530	188.25
20×40_2(2)	17440[a]	-[b]	15462	15600	107.91	15829	16290	172.95
20×40_2(3)	17240[a]	-[b]	15423.5	15540	104.45	15576	15850	179.65
20×40_2(4)	16700[a]	-[b]	14856.5	15060	104.26	15653.5	15780	163.25
20×40_2(5)	16690[a]	-[b]	15034	15250	99.74	15049	15550	177.85
30×60_2(1)	20540[a]	-[b]	19346.5	19530	389.06	19535	19790	609.3
30×60_2(2)	21060[a]	-[b]	19695	19830	477.57	20444.5	20570	587.7
30×60_2(3)	20710[a]	-[b]	19375.5	19550	390.22	19580.5	19860	626.1
30×60_2(4)	21890[a]	-[b]	20806.5	20910	379.60	21202.5	21370	575
30×60_2(5)	21170[a]	-[b]	19749.5	20010	359.04	20241.5	20560	559.3
30×60_1(1)	-[a]	-[b]	23498.5	23940	387.13	23363	24070	843.3
30×60_1(2)	25180[a]	-[b]	22782.5	23200	487.88	24272.5	24350	727.9
30×60_1(3)	-[a]	-[b]	22959	23270	418.27	23042	23570	661.9
30×60_1(4)	-[a]	-[b]	23583.5	23840	384.61	22849	23570	699
30×60_1(5)	24480[a]	-[b]	22793.5	23180	394.47	22983.5	23450	685.5

[a] *The best known solution obtained within 24 hours*
[b] *CPU time = 24 hours*

A solution of 10×20_1(1) is represented in Appendix *E*. The LR-SA can reach the optimal solutions for most of instances in Group I data sets. On the run of LR-SA, the average CPU time required is longer than that of SA-TS. It is due to the algorithm applied in the

construction phase. The Lagrangian relaxation procedure takes longer time to generate the initial solutions. Nevertheless, better solutions on the objective function values are obtained.

Table 6.4 Comparison of the algorithm results and the optimal solutions on Group I data sets

Data Set	Algorithm SA-TS		Algorithm LR-SA	
	$\Phi_{(1)}^{SA-TS}$ (%)	$\Phi_{(2)}^{SA-TS}$ (%)	$\Phi_{(1)}^{LR-SA}$ (%)	$\Phi_{(2)}^{LR-SA}$ (%)
5×5_1(1)	2.65	2.65	0.00	0.00
5×5_1(2)	0.14	0.00	2.29	0.00
5×5_1(3)	0.00	0.00	0.00	0.00
5×5_1(4)	0.78	0.00	0.00	0.00
5×5_1(5)	0.00	0.00	0.00	0.00
5×5_2(1)	4.26	3.91	3.91	3.91
5×5_2(2)	0.00	0.00	0.00	0.00
5×5_2(3)	1.17	0.00	0.00	0.00
5×5_2(4)	3.49	2.52	2.52	2.52
5×5_2(5)	0.00	0.00	0.00	0.00
10×10_1(1)	0.64	0.00	0.00	0.00
10×10_1(2)	0.77	0.00	0.00	0.00
10×10_1(3)	0.00	0.00	0.00	0.00
10×10_1(4)	0.05	0.00	0.00	0.00
10×10_1(5)	0.48	0.48	0.00	0.00
10×10_2(1)	0.51	0.35	0.35	0.35
10×10_2(2)	0.58	0.00	0.00	0.00
10×10_2(3)	0.09	0.00	0.13	0.00
10×10_2(4)	2.42	1.57	1.97	1.97
10×10_2(5)	2.04	1.92	1.92	1.92
15×15_1(1)	0.36	0.32	0.00	0.00
15×15_1(2)	0.00	0.00	0.00	0.00
15×15_1(3)	0.35	0.00	0.00	0.00
15×15_1(4)	2.06	0.30	0.00	0.00
15×15_1(5)	1.65	0.32	0.00	0.00
15×15_2(1)	1.69	1.20	1.20	1.20
15×15_2(2)	0.87	0.70	0.00	0.00
15×15_2(3)	0.22	0.00	0.00	0.00
15×15_2(4)	0.91	0.46	0.89	0.46
15×15_2(5)	2.18	0.76	1.04	1.02
20×20_1(1)	0.02	0.00	0.00	0.00
20×20_1(2)	2.54	1.85	0.00	0.00
20×20_1(3)	1.05	0.46	0.00	0.00
20×20_1(4)	0.00	0.00	0.00	0.00
20×20_1(5)	2.53	0.92	0.00	0.00
20×20_2(1)	1.51	0.88	1.27	0.71
20×20_2(2)	3.06	1.75	0.98	0.70
20×20_2(3)	4.98	2.95	3.78	2.60
20×20_2(4)	1.39	0.51	1.79	0.85
20×20_2(5)	2.10	1.06	1.39	0.88
Average	1.24	0.70	0.64	0.48
Maximum	4.98	3.91	3.91	3.91
Minimum	0.00	0.00	0.00	0.00

Table 6.5 Comparison of the algorithm results and the optimal solutions on Group II data sets

Data Set	Algorithm SA-TS		Algorithm LR-SA	
	$\Phi_{(1)}^{SA-TS}$ (%)	$\Phi_{(2)}^{SA-TS}$ (%)	$\Phi_{(1)}^{LR-SA}$ (%)	$\Phi_{(2)}^{LR-SA}$ (%)
10×20_1(1)	5.86	4.48	3.36	3.16
10×20_1(2)	7.88	5.97	7.13	5.72
10×20_1(3)	7.32	6.13	8.01	4.95
10×20_1(4)	8.63	6.48	6.01	5.69
10×20_1(5)	4.75	3.97	2.67	2.05
10×20_2(1)	6.19	3.92	3.54	3.13
10×20_2(2)	9.72	8.49	6.70	6.25
10×20_2(3)	11.76	10.16	7.65	6.90
10×20_2(4)	7.98	6.73	5.43	4.79
10×20_2(5)	7.47	6.51	6.03	6.00
20×30_1(1)	4.01	3.00	2.40	2.18
20×30_1(2)	3.78	1.97	1.80	1.62
20×30_1(3)	5.93	4.79	3.74	3.74
20×30_1(4)	7.65	6.32	4.47	3.39
20×30_1(5)	4.88	3.37	2.58	1.51
20×30_2(1)	5.09	3.90	4.86	4.74
20×30_2(2)	4.86	3.80	2.56	2.56
20×30_2(3)	10.05	8.57	5.53	4.61
20×30_2(4)	7.75	7.08	6.00	5.96
20×30_2(5)	9.55	7.73	6.77	4.98
20×40_1(1)	3.83	2.20	2.14	1.59
20×40_1(2)	9.68	7.99	5.38	4.93
20×40_1(3)	6.09	4.79	5.33	3.37
20×40_1(4)	7.33	6.28	7.10	5.77
20×40_1(5)	8.93	7.36	4.37	3.50
20×40_2(1)	7.61	6.61	4.36	4.08
20×40_2(2)	11.34	10.55	9.24	6.59
20×40_2(3)	10.54	9.86	9.65	8.06
20×40_2(4)	11.04	9.82	6.27	5.51
20×40_2(5)	9.92	8.63	9.83	6.83
30×60_2(1)	5.81	4.92	4.89	3.65
30×60_2(2)	6.48	5.84	2.92	2.33
30×60_2(3)	6.44	5.60	5.45	4.10
30×60_2(4)	4.95	4.48	3.14	2.38
30×60_2(5)	6.71	5.48	4.39	2.88
30×60_1(1)	-	-	-	-
30×60_1(2)	9.52	7.86	3.60	3.30
30×60_1(3)	-	-	-	-
30×60_1(4)	-	-	-	-
30×60_1(5)	6.89	5.31	6.11	4.21
Average	7.41	6.13	5.17	4.24
Maximum	11.76	10.55	9.83	8.06
Minimum	3.78	1.97	1.80	1.51

We calculate the deviation of the best and the average objective function values of the proposed algorithms from best known/optimal value, denoted as $\Phi_{(1)}^{LR-SA}$ and $\Phi_{(2)}^{LR-SA}$ as shown in Tables 6.4 and 6.5.

For Group I data sets, the maximum value of $\Phi_{(1)}^{LR-SA}$ is less than 4%, which is better than the value obtained by Algorithm SA-TS (4.98%). Similar to the performance of SA-TS, Algorithm LR-SA is able to obtain the optimal solutions for all instances in data sets 5×5_1, 10×10_1, 15×15_1 and 20×20_1. The average values of the deviation of the best and the average objective function values also decrease from 1.24% to 0.64% and 0.70% to 0.48%, respectively.

LR-SA also shows good performance in Group II data sets. The worst values of $\Phi_{(1)}^{LR-SA}$ and $\Phi_{(2)}^{LR-SA}$ are less than 10%. For large instances, such as 30×60_1 and 30×60_2, the values are much better compared against those of SA-TS.

6.4 Conclusions

In this chapter, we have studied and presented a successful hybrid algorithm, Algorithm LR-SA, for tackling the TACS problem. The algorithm integrates the Lagrangian relaxation procedure and Simulated Annealing algorithm. The performance of the proposed algorithm is measured by applying the algorithm to the same problems presented in the earlier chapters.

The computational results show that the proposed algorithm is able to produce good quality solutions which are less than 10% of the best known or optimal solutions. For most problem instances in Group I data sets, the optimal solutions can be reached by the proposed algorithm. We conclude that Algorithm LR-SA outperforms Algorithm SA-TS proposed in Chapter 6 in terms of both the average and the best objective function values obtained. The Lagrangian relaxation approach is able to construct good quality initial solutions that would lead to better quality final solutions.

This work has raised several issues, which remain as possible future research directions. Some possible extensions of the proposed mathematical model can be explored, such as by incorporating specific constraints or requirements of other universities or institutions, incorporating the classroom assignment sub-problem and so forth. Finally, we extend the idea of the hybrid algorithm to solve another category of the educational timetabling problem, namely, the examination timetabling problem. The details would be explained in Chapter 7.

CHAPTER VII
HYBRIDIZATION OF METAHEURISTICS FOR THE EXAMINATION TIMETABLING PROBLEM

7.1 Introduction

The examination timetabling problem is another category of the educational timetabling problem. Setting up a conflict-free timetable is not a trivial task due to limited resources like periods and examination rooms. In particular, the primary goal in the examination timetabling problem is no student takes more than one examination in any time period. This conflict is categorized as a hard constraint and must be eliminated.

Several authors define a situation when a student is required to attend more than one examination at the same time as the first-order conflict. (Balakrishnan, et al., 1992 and Bullnheimer, 1998). Besides the first-order conflicts, there are so-called back-to-back or second-order conflicts. This term refers to a situation where a student has to take two consecutive examinations. Finally, there may be higher-order conflicts dealing with other constraints, such as room capacities and so on.

Different variants of the examination timetabling problem have been proposed in the literature. The allocation is an important and difficult task for each educational institutions since it requires expensive human and computer resources. Several surveys of practical applications of examination timetabling algorithms were discussed by Carter (1986) and Burke et al. (1996). The following soft constraints represent some of the common requirements in the examination timetabling problem:

(1) Limitation on available rooms per time period.

(2) Students should not have examinations in consecutive periods or more than one examination on the same day.

The examination timetabling problem can be represented as an undirected weighted graph, where vertices and edges represent examinations and conflicts between examinations, respectively (de Werra, 1985). For illustration purpose, Figure 7.1 shows a simple examination problem. For example, the weight of course 1 is 20, meaning that 20 students are enrolled in this course. The number of students who are enrolled in examinations 1 and 3 is 4 students.

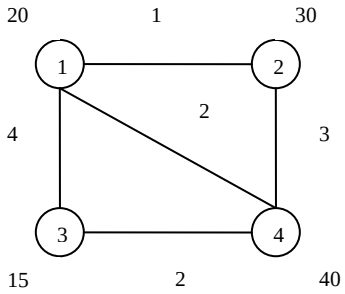


Figure 7.1 An example of examination timetabling problem as an undirected weighted graph

The above graph is linked to the graph colouring problem. The problem is an assignment of colour to vertices in such a way that no two adjacent vertices share the same colour. The graph colouring problem is known as NP-complete (Garey and Johnson, 1977).

Although a graph theory approach is commonly used to generate the examination schedule, various restrictions introduced to the problem, such as the second order and higher-order conflicts, limit the use of a graph theoretic approach (Arani and Lotfi, 1989). To deal with these conflicts, Leong and Yeong (1990) have formulated the examination timetabling problem as Quadratic Assignment Problem (QAP).

The rest of this chapter is organized as follows. Section 7.2 describes a review of QAP and its relationship to the examination timetabling problem. A hybrid algorithm is then proposed in order to solve the problem, namely, Algorithm GRASP-SA-TS. Basically, three different algorithms involved are: GRASP (Greedy Randomized Adaptive Search Procedure), Simulated Annealing (SA) and Tabu Search (TS).

The idea of the hybrid algorithm is similar to that of Algorithm SA-TS proposed in Chapter 5. The main difference lies in the algorithm implemented in the construction phase. Here, we propose GRASP algorithm in order to build an initial solution in the construction phase. The computational results obtained with our hybrid algorithm are reported. Section 7.3 presents the extended examination timetabling problem, including the mathematical programming model, the lower bound, the proposed algorithm and the computational results as well. Finally, some concluding remarks are provided in Section 7.4.

7.2 The Quadratic Assignment Problem (QAP)

The QAP was first introduced by Koopmans and Beckmann (1957) to model a plant location problem. This problem belongs to the class of NP-hard combinatorial optimization problems (Sahni and Gonzales, 1976). Due to wide assortment of applications, The QAP has been extensively used to formulate other practical problems as summarized in Table 7.1.

Some of the last surveys about QAP in the literature were presented by Burkard (1984), Burkard (1990), Anstreicher (2003), Drezner et al. (2005) and Loiola et al. (2007). The article of Anstreicher (2003) reviewed the recent advances in the solutions of QAP. A summary of the progress made in both heuristic and exact solutions for the QAP was presented by Drezner et al. (2005). By configuring some new QAP instances with non-uniform distributed flow

distances and flows, the authors showed that those instances are difficult for metaheuristics. Loiola et al. (2007) continued to summarize new publications about QAP since 1999 including some of the most important formulations as well as a detailed discussion on the theoretical resources used to define lower bounds for exact and heuristic solutions methods.

Table 7.1 Several applications of the Quadratic Assignment Problem

No	Problem	Authors
1	Economic problems	Koopmans and Beckmann (1957) and Heffley (1980)
2	Scheduling problems	Geoffrion and Graves (1976)
3	Facility layout problems	Elshafei (1977), Dickey and Hopkins (1972), Benjaafar (2002) and Miranda et al. (2005)
4	Timetabling problems	Leong and Yeong (1987), Arani and Lotfi (1989) and Bullnheimer (1998)
5	Electronic component placement problems	Steinberg (1961) and Miranda et al. (2005)

The QAP is identified as the problem of finding a minimum cost allocation of facilities into locations, taking the costs as the sum of all possible distance-flow products. This problem can be formulated as integer programming (IP) formulations (Wilhelm and Ward, 1987; Rossin et al., 1999 and Fedjki et al., 2004). In these formulations, the quadratic terms are included in the objective function value. The QAP can also be formulated as mixed integer linear programming (MILP) formulations by transforming quadratic terms into linear terms (Lawler, 1963; Frieze and Yadegar, 1983 and Adam and Johnson, 1994).

Taking a simple approach, the pairwise allocation of facility costs to adjacent locations is proportional to flows and the distances between them. In this case, the QAP can be formulated by using the permutation concept (Hillier and Connors, 1966; Taillard, 1991 and Lim et al., 2000) as follows: the objective is to find a permutation of n facilities such that the total cost $C(\pi)$ is minimized. Let f_{ij} be the flow between facilities i and j and $d_{\pi(i)\pi(j)}$ be the distance between locations $\pi(i)$ and $\pi(j)$. The QAP problem then becomes:

$$\min_{\pi \in \Pi(n)} C(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\pi(i)\pi(j)} \tag{7.1}$$

where $\Pi(n)$ is the set of all permutations of integers {1, 2, …, *n*}.

Here, the solution is represented by the *n*-vector: $\pi = [\pi(1), \pi(2), \pi(3), \ldots, \pi(n)]$ and the element $\pi(i) = k$ denotes that facility *i* is assigned to location *k* in the current solution $\pi \in \Pi(n)$.

The permutation concept is similar to that of the permutation scheduling problem (PFSP). PFSP is a class of scheduling problems in which the operations of every job must be processed on machines in the same order and the processing order of the jobs on the machines is the same for every machine (Tseng and Lin, 2009). In PFSP, the objective is to find a permutation of *n* jobs, $\pi = \{\pi(1), \pi(2), \ldots, \pi(n)\}$, in the set of all permutations $\Pi(n)$ such that the total flow time (makespan) is minimized.

The quality of the solutions obtained from heuristic algorithms is measured by the gap between the lower bound value and the optimal solution. One of well-known QAP lower bound was presented by Gilmore (1962) and Lawler (1963), namely, GLB. Its importance is due to its simplicity. However, it shows an important drawback as its gap grows very quickly with the size of the problem.

Drezner (1995) proved that bound based on the linear programming relaxation is equal or better than the GLB bound. Other different types of lower bounds proposed are the elimination bound (ELI), an interior point based linear programming bound (IPLP) (Resende et al., 1995), a semi-definite programming bound (SDP) (Rendl and Sotirov, 2007) and a triangle decomposition bound (TDB) (Karisch and Rendl, 1995).

In the literatures, two different approaches for solving the QAP are divided into two categories: exact and heuristic algorithms. In the first case, the most frequent used strategies are branch and bound, branch and cut and dynamic programming techniques. As exact methods can only be used to solve small-size instances of the problem, much research effort has been devoted to the development of heuristic solution procedures that can be performed in reasonable computation time and yet yield optimal or near-optimal solutions.

Heuristic algorithms are classified into the following categories: constructive, limited enumeration and improvement methods/metaheuristics (Loiola et al., 2007). There are number of heuristic techniques using different conceptions as summarized in Table 7.2. Constructive methods construct a sub-optimal permutation step by step. It starts with a partial permutation π which is empty. Then π is expanded by repetitively selecting a pair of assignment (i, j) such that $i \notin M$ and $j \notin \pi(M)$ according to certain heuristic, where M is the index set containing the indices of π for which the corresponding assignments are done and $\pi(M)$ is the set $\{\pi(i) \mid i \in M\}$. This process is repeated until π becomes a complete permutation.

Enumeration methods can guarantee the optimal solution only if they can go to the end of the enumerative process. Unfortunately, it usually takes much longer to reach the optimality. In order to limit the computation time of the enumeration, stopping conditions are defined: maximum number of iterations, a limit for the execution time and so forth.

Improvement methods are mostly used in the QAP. It is started with a feasible solution that will be further improved by searching for possible solutions in its neighborhood. In particular, improvement heuristic methods are frequently used in metaheuristics.

Table 7.2 Some applications of heuristics to the Quadratic Assignment Problem

No	Methods	Authors
1	Constructive methods	Arkin et al. (2001), Gutin and Yeo (2002) and Yu and Sarker (2003)
2	Enumeration methods	Burkard and Bonniger (1983) and West (1983)
3	Simulated Annealing	Burkard and Rendl (1984), Connoly (1990), Peng et al. (1996), Tian et al. (1996) and Siu and Chang (2002)
4	Genetic Algorithm	Ahuja et al. (2000), Lim et al. (2000) and Lim et al. (2002)
5	Ant Colony Algorithm	Dorigo et al. (1996) and Maniezzo and Colorni (1995, 1999)
6	Tabu Search	Kapov (1990), Taillard (1991) and Drezner (2005)
7	GRASP	Li et al. (1994)

As mentioned in Section 7.1, there is a strong relationship between Quadratic Assignment and the examination timetabling problem. Examinations can be treated as facilities that need to be scheduled on different time periods (locations). Here, we formulate the basic examination timetabling problem as a Quadratic Assignment Problem by assuming that the number of examinations and available time periods are equal. Each examination has to be scheduled on one time period and each time period can only be occupied by one examination. These requirements are identical with the requirements of the QAP.

In QAP, the objective function represents the total cost of assignment of all facilities to all locations, which is the product of the flow between facilities and the distance between locations. The primary goal in the examination timetabling problem is to avoid overlap of examinations having common students. Other goals such as to minimize the number of students who have two examinations in any consecutive periods and to spread examinations with higher number of common students as much as possible can also be considered. Thus,

these goals can be formulated using the objective function in the QAP with appropriate cost values that would be explained in the next section.

The following sub-sections will describe the mathematical programming model of the basic examination timetabling problem, the proposed algorithm as well as the computational results in details.

7.2.1 The Mathematical Programming Model

Consider the basic examination scheduling problems with n (= $|M|$) slots or time periods and n (= $|M|$) examinations to be scheduled. Given two matrices $FLOW = [f'_{tu}]$ and $COST = [c'_{vw}]$ where elements f'_{tu} and c'_{vw} represent the number of students taking examinations t and u and the cost between time period v and time period w, respectively, the examination problem can be formulated as a Quadratic Assignment Problem (EP Model I).

We define the following decision variable:

- x_{tv} = 1 if exam t is scheduled at time period v, 0 otherwise $\quad (1 \le t,v \le n)$

[EP Model I] (Leong and Yeong, 1987)

Minimize $$Z_{EP\_1} = \sum_{t=1}^{n}\sum_{u=1}^{n}\sum_{v=1}^{n}\sum_{w=1}^{n} f'_{tu}c'_{vw}x_{tv}x_{uw} \tag{7.2}$$

subject to:

$$\sum_{t=1}^{n} x_{tv} = 1 \qquad (1 \le v \le n) \tag{7.3}$$

$$\sum_{v=1}^{n} x_{tv} = 1 \qquad (1 \le t \le n) \tag{7.4}$$

$$x_{tv} \in \{0,1\} \qquad (1 \le t,v \le n) \tag{7.5}$$

The value of c'_{vw} is then calculated as follows:

$$c'_{vw} = \begin{cases} 1/d_{vw} & v \neq w \\ \hat{M} & v = w \end{cases} \tag{7.6}$$

where d_{vw} = the "distance" between time periods *v* and *w*

$\hat{M}$ = a very large number

The following illustration explains how to determine and calculate the value of c'_{vw}. Linear distance cost which is generally used in QAP cannot be applied directly to the examination timetabling problem. Here, we use a cost function that is inversely proportional to distance. Let assume that the number of days per week is 5 days and the number of time periods per week is equal to 40 periods. The distance between time period 3 and time period 9 is 6 time periods and the cost between these time periods c'_{39} is $1/6$.

In order to avoid a situation where students have to take more than one examination at one time period (first-order conflict), the cost between time periods *v* and *w* (when *v* = *w*) is set to a very large number. The main objective in this model is to minimize the number of students who have two examinations in any consecutive periods (second-order conflict), as shown in equation (7.2). Both constraints (7.3) and (7.4) ensure that one time period can only be occupied by one examination and each examination can only be scheduled on one time period and, respectively. These constraints also ensure that the first-order conflict will not occur.

The EP Model I can be represented as a permutation problem:

$$\min_{\pi \in \Pi(n)} C(\pi) = \sum_{t=1}^{n} \sum_{u=1}^{n} f'_{tu} c'_{\pi(t)\pi(u)} \tag{7.7}$$

where $\Pi(n)$ is the set of all permutations of integers {1, 2, …, *n*}.

The solution is represented by the *n*-vector: $\pi = [\pi(1), \pi(2), \pi(3), \ldots, \pi(n)]$ and the element $\pi(t) = r$ denotes that examination *t* is assigned to time period *r* in the current solution $\pi \in \Pi(n)$.

7.2.2 The Proposed Algorithm

In this chapter, we introduce a new hybrid metaheuristic for the examination timetabling problem, which involves GRASP as well as a hybridization of SA and TS algorithms as presented in Chapter 5. The Greedy Randomized Adaptive Search Procedure (GRASP) is a simple metaheuristic that combines constructive heuristics and a local search (Feo and Resende, 1995). Several applications of GRASP can be found in the following problems: assignment problems (Prais and Riberio, 2000), the Quadratic Assignment Problem (Yong et al., 1994), airline flight scheduling and maintenance base planning (Feo and Bard, 1989) and scheduling of parallel machines (Laguna and Velarde, 1991).

The entire algorithm comprises of two main phases: (1) construction, and (2) improvement. The GRASP is used to initialize a solution in the first phase, while a combination of SA and TS (denoted as the Algorithm SA-TS) is for improving the solution in the second phase. Each phase is presented and described in the following sub-sections.

7.2.2.1 The Construction Phase

In the construction phase, we build an initial solution by implementing part of the Greedy Randomized Adaptive Search Procedure (GRASP) (Yong et al., 1994). GRASP consists of two processes: a construction process and a local search process. In the first phase (construction phase), an initial feasible solution is constructed. Since the solutions by a GRASP construction are not guaranteed to be optimal, it is almost always beneficial to apply a

local search to attempt to improve the initial solution in the local search process. Normally, a local optimization procedure such as a two-exchange is employed. However, in our implementation, we only consider the first process of GRASP in order to build an initial solution. The initial solution is then improved by Algorithm SA-TS proposed in Chapter 5.

Construction Phase ( )

(1) Sort the $(n^2-n)/2$ flow entries in *FLOW* in increasing order and keep the largest $\lfloor \beta(n^2-n)/2 \rfloor$ entries, such that $f'_{t_1u_1} \geq f'_{t_2u_2} \geq \ldots \geq f'_{t_{\lfloor \beta(n^2-n)/2 \rfloor}u_{\lfloor \beta(n^2-n)/2 \rfloor}}$. Let β be the first candidate restriction parameter $(0 < \beta < 1)$ and $\lfloor x \rfloor$ be the largest integer smaller or equal to *x*.

(2) Sort the $(n^2-n)/2$ cost entries in *COST* in non-increasing order and keep the smallest $\lfloor \beta(n^2-n)/2 \rfloor$ entries, such that $c'_{v_1w_1} \leq c'_{v_2w_2} \leq \ldots \leq c'_{v_{\lfloor \beta(n^2-n)/2 \rfloor}w_{\lfloor \beta(n^2-n)/2 \rfloor}}$.

(3) Calculate the cost interactions $f'_{t_1u_1}c'_{v_1w_1}, f'_{t_2u_2}c'_{v_2w_2}, \ldots, f'_{t_{\lfloor \beta(n^2-n)/2 \rfloor}u_{\lfloor \beta(n^2-n)/2 \rfloor}}c'_{v_{\lfloor \beta(n^2-n)/2 \rfloor}w_{\lfloor \beta(n^2-n)/2 \rfloor}}$, sort them in increasing order and keep the smallest $\lfloor \gamma\beta(n^2-n)/2 \rfloor$ elements as the *candidate list*, where γ is the second candidate restriction parameter $(0 < \gamma < 1)$.

(4) Select a couple of assignment pairs from the *candidate list* randomly.

(5) Calculate C_{tv}, the cost of assigning examination *t* to time period *v*, with respect to the already-made assignments, Γ:

$$C_{tv} = \sum_{(u,w)\in\Gamma} f'_{tu}c'_{vw} \text{ where } \Gamma = \{(u_1,w_1),(u_2,w_2),\ldots,(u_{\tilde{r}},w_{\tilde{r}})\}$$

(6) Set *o* = the number of unassigned examinations and *m* = the number of unassigned examination-time period pairs

(7) Determine the $\lfloor \gamma m \rfloor$ examination-time period pairs having the smallest C_{tv} values.

(8) Select an examination-time period pair (t,v) randomly from the list generated in Step 7.

(9) Update the set $\Gamma = \Gamma \cup (t,v)$

(10) Set $o = o - 1$

(11) Repeat Steps 5 – 10 until *o* = 0

Figure 7.2 Construction phase of GRASP Algorithm

The process is started by selecting the first 2 assignments based on the minimum cost of interaction $f'_{tu}c'_{vw}$, followed by assigning the remaining (n – 2) examinations based on the cost of assigning a particular examination with respect to the already-made assignments, i.e. we select the one that has the minimum cost. This process is made until all the remaining (n – 2) examinations are assigned. The time complexity for finding minimum cost of interaction $f'_{tu}c'_{vw}$ is $O(n^2)$. The details of the construction process of GRASP are described in Figure 7.2 (Yong et al., 1994).

A numerical example for illustrating the construction phase in the GRASP algorithm is given as follows. Consider the basic examination scheduling problems with n (= 5) time periods and n (= 5) examinations to be scheduled. Two matrices $FLOW = [f'_{tu}]$ and $COST = [c_{vw}]$ represent the number of students who take examinations and the cost between two different time periods, respectively. For example, there are 10 students who take examinations 1 and 2 as represented by f'_{12} and the cost between time periods 1 and 3 is 0.5.

FLOW

	1	2	3	4	5
1	0	10	0	15	5
2		0	2	0	25
3			0	7	8
4				0	2
5					0

COST

	1	2	3	4	5
1	-	1	0.5	0.33	0.25
2		-	1	0.5	0.33
3			-	1	0.5
4				-	1
5					-

Figure 7.3 A numerical example for illustrating GRASP algorithm

Let β = 0.5 and γ = 0.6, we sort $(n^2 - n)/2$ = 10 flow entries in *FLOW*, keeping the $\lfloor \beta(n^2 - n)/2 \rfloor$ = 5 largest values: $f'_{25} \geq f'_{14} \geq f'_{12} \geq f'_{35} \geq f'_{34}$ and sort $(n^2 - n)/2$ = 10 cost entries in *COST*, keeping the $\lfloor \beta(n^2 - n)/2 \rfloor$ = 5 smallest values: $c'_{15} \leq c'_{14} \leq c'_{25} \leq c'_{13} \leq c'_{24}$. Then, we sort the costs of interaction $f'_{25}c'_{15}, f'_{14}c'_{14}, f'_{12}c'_{25}, f'_{35}c'_{13}, f'_{34}c'_{24}$ in increasing order and keep the smallest $\lfloor \gamma\beta(n^2 - n)/2 \rfloor$ = 3 elements: $f'_{12}c'_{25} \leq f'_{34}c'_{24} \leq f'_{35}c'_{13}$ as the candidate list (Figure 7.4).

$f'_{12}c'_{25}$	$f'_{34}c'_{24}$	$f'_{35}c'_{13}$	$f'_{14}c'_{14}$	$f'_{25}c'_{15}$
3.33	3.5	4	5	6.25

Figure 7.4 A numerical example of the costs of interaction in GRASP algorithm

We then select a couple of assignments having the smallest interaction cost from the candidate list randomly. Note that the above ordering of the cost elements needs to be done only once in the initialization phase of the GRASP. Assuming that $f'_{12}c'_{25}$ is selected, meaning that courses 1 and 2 are scheduled on time periods 2 and 5, respectively, we continue to schedule unscheduled examinations to one time period at a time.

Let Γ = {(1,2), (2,5)} be the set of already-made assignment, we calculate C_{tv}, the cost of assigning examination t to time period v, with respect to the already-made assignment, Γ, as follows: C_{31}, C_{33}, C_{34}, C_{41}, C_{43}, C_{44}, C_{51}, C_{53}, C_{54} (Figure 7.5). We sort these values in increasing order and keep the smallest 5 elements: C_{31}, C_{33}, C_{34}, C_{44}, C_{51}. We then select one element randomly. Assuming that C_{34} is selected, we update the set $\Gamma = \Gamma \cup \{(3,4)\}$. This process in continued until all unscheduled examinations have been scheduled.

C_{31}	C_{33}	C_{34}	C_{44}	C_{51}	C_{41}	C_{43}	C_{53}	C_{54}
0.5	1	2	7.5	11.25	15	15	17.5	27.5

Figure 7.5 A numerical example of C_{tv} in GRASP algorithm

7.2.2.2 The Improvement Phase

The quality of the initial solution generated by GRASP, *initial_sol*, is then improved in the improvement phase. The algorithm applied in this phase is a combined SA and TS algorithm (Algorithm SA-TS). While it is mainly based on Simulated Annealing (Kirkpatrick et al, 1983), the main difference of the standard SA and the proposed SA lies in the additional elements or strategies added. Several features from Tabu Search, such as the tabu length, tabu list and the intensification strategy are incorporated in the algorithm for further improvement (Glover, 1989). The details of this procedure are summarized in Figure 7.6.

Instead of selecting two examinations randomly as was commonly done in Simulated Annealing, we start by selecting one examination t randomly followed by examining all other potential pair-swaps sequentially in the order $\{(t,u): u \neq t\}$. The selected move is the one with the best $\Delta(\pi,t,u)$ value. Assuming that $f'_{tt} = f'_{uu} = 0$, the objective function difference $\Delta(\pi,t,u)$

obtained by exchanging examinations $\pi(t)$ and $\pi(u)$ can be computed in $O(n)$ operations (Taillard and Gambardella, 1997):

$$\begin{aligned}\Delta(\pi,t,u) \quad = \quad & f'_{tu}\left(c'_{\pi(u)\pi(t)} - c'_{\pi(t)\pi(u)}\right) + f'_{ut}\left(c'_{\pi(t)\pi(u)} - c'_{\pi(u)\pi(t)}\right) + \\ & \sum_{\substack{a=1 \\ a\neq t,u}}^{n} \{ f_{at}\left(c'_{\pi(a)\pi(u)} - c'_{\pi(a)\pi(t)}\right) + f_{au}\left(c'_{\pi(a)\pi(t)} - c'_{\pi(a)\pi(u)}\right) + \\ & f_{ta}\left(c'_{\pi(u)\pi(a)} - c'_{\pi(t)\pi(a)}\right) + f_{ua}\left(c'_{\pi(t)\pi(a)} - c'_{\pi(u)\pi(a)}\right) \}\end{aligned} \tag{7.8}$$

Given two matrices $FLOW = [f'_{tu}]$ and $COST = [c'_{vw}]$ where elements f'_{tu} and c'_{vw} represent the number of students taking examinations t and u and the cost between time period v and time period w, respectively, if both matrices $FLOW = [f'_{tu}]$ and $COST = [c'_{vw}]$ are symmetric with a zero diagonal, the formula can be simplified as follows:

$$\Delta(\pi,t,u) \quad = \quad 2\times \sum_{\substack{a=1 \\ a\neq t,u}}^{n} f'_{at}\left(c'_{\pi(a)\pi(u)} - c'_{\pi(a)\pi(t)}\right) + f'_{au}\left(c'_{\pi(a)\pi(t)} - c'_{\pi(a)\pi(u)}\right) \tag{7.9}$$

The new permutation is then evaluated by the acceptance-rejection procedure in SA. The tabu list contains pairs (t, u) that have been visited in the last *length* iterations. For a given iteration, if a pair (t, u) belongs to the tabu list, it is not allowed to accept the exchange of examinations t and u, unless this exchange gives an objective function value strictly better than the best solution obtained so far. At any temperature T, the neighborhood search is repeated until a certain number of iterations, *inner_loop*, has been performed.

If there is no improvement of the solution obtained after a certain number of iterations (*limit*), we apply the intensification strategy of Tabu Search. This strategy focuses the search once again starting from the best permutation obtained. Finally, the entire algorithm will be terminated if the total number of iterations of the outer loop reaches the preset maximum number of iterations, *outer_loop*.

Algorithm SA-TS ( )
(1) Initialize the parameters
(2) Set the best solution, *best_sol* = *initial_sol*
(3) Set the current solution, *current_sol* = *initial_sol*
(4) Set the total number of iterations, *num_iter* = 0
(5) Set the total number of iterations without improvement, *no_improv* = 0
(6) **While** the total number of iterations, *num_iter* is less than the preset maximum number of iterations, *outer_loop* do:
(7) Repeat *inner_loop* times:
(8) Select an examination *t* randomly
(9) Apply a partial sequential neighborhood search
(10) Find the best permutation π' with the smallest value of $\Delta(\pi',t,u)$
(11) **If** $\Delta(\pi',t,u) < 0$
(12) Update the current solution, *current_sol*
(13) Update tabu list
(14) **If** *current_sol* is better than *best_sol*
(15) Update the best solution, *best _sol* = *current_sol*
(16) **Else**
(17) Choose a random number *r* uniformly from [0,1]
(18) *no_improv* := *no_improv* + 1
(19) Check whether the best permutation is tabu or not
(20) **If** $r < exp^{-\Delta(\pi',t,u)/T_{num\_iter}}$ and the new solution is not tabu
(21) Update the current solution, *current_sol*
(22) Update tabu list
(23) **Else**
(24) Return to the current solution, *current_sol*
(25) Update tabu list
(26) Update temperature $T_{num\_iter} := \alpha T_{num\_iter}$
(27) **If** (*no_improv* > *limit*)
(28) Apply the intensification strategy
(29) Set *no_improv* := 0
(30) *num_iter* := *num_iter* +1
(31) **End while**
(32) Report the best solution, *best_sol*

Figure. 7.6 Algorithm SA-TS for the basic examination timetabling problem

7.2.3 Computational Results

As mentioned in previous sections, the examination timetabling problem is presented as a Quadratic Assignment Problem. In order to evaluate the performance of the proposed algorithm, we decided to solve some benchmark problems from a library for research on the QAP (QAPLIB http://www.opt.math.tu-graz.ac.at/qaplib/inst.html) which have been studied and solved by other researchers (Burkard et al., 1997). In this section, the computational

results of applying the proposed algorithm to solve some benchmark problems of the QAPLIB are shown. We report experimental results on a wide range of test problems from the suite of QAP test problem QAPLIB. We tested the proposed algorithm implementation on several problem classes in QAPLIB: *chr*, *had*, *kra*, *nug*, *rou*, *scr*, *sko*, *tai*, and *wil* classes.

The computational experiments were performed on a 2.67 GHz Intel (R) Core^{TM} 2 Duo CPU with 3 GB of RAM under the Microsoft Windows Vista Operating System. The proposed algorithm coded in C++. The values of the parameters used in the computational study are summarized in Table 7.3. All the values are determined experimentally to ensure a compromise between the computation time and the solution quality.

Table 7.3 Parameter settings for Algorithm SA-TS (the basic examination timetabling problem)

Parameter	Value
Maximum number of iterations, *outer_loop*	$300\times n$
Initial temperature, T_0	5,000
Number of neighborhood moves at each temperature *T*, *inner_loop*	$100\times n$
Cooling factor, α	0.9
Number of non-improvement iterations prior to intensification, *Limit*	$0.02\times$*outer_loop*
Length of tabu list, *length*	$n/2$

For each benchmark problem, the proposed algorithm was executed 20 times with different random seeds. The following tables summarize the average objective function value obtained, the best objective function value obtained and the average CPU time required to obtain the solution for each class.

The objective function values of the optimal/best known solutions given in Burkard et al. (1997) are presented for comparison purposes. The heading $\Phi_{(1)}^{SA-TS}$ refers to the percentage deviation between the average objective function value of the solutions obtained and the best

known/optimal solution, while $\Phi_{(2)}^{SA-TS}$ refers to the percentage deviation between the best objective function value of the solutions obtained and the best known/optimal solution. The values for $\Phi_{(1)}^{SA-TS}$ and $\Phi_{(2)}^{SA-TS}$ are computed by equations which are similar to equations (4.4) and (4.5).

Table 7.4 summarizes the computational results of the *chr* type benchmarks. The difficulty level in solving the *chr* problem instances is considered high (Lim et al., 2002). On the whole, the proposed hybrid algorithm is able to find solutions with values of $\Phi_{(1)}^{SA-TS}$ not exceeding 1.50% from the known optimum. For all problem instances, the best known/optimal solutions can be obtained.

Table 7.4 Computational results of Algorithm SA-TS on problem class *chr*

Benchmark problem	Optimal/Best known solution	Average Solution	Average CPU time (seconds)	Best Solution	$\Phi_{(1)}^{SA-TS}$ (%)	$\Phi_{(2)}^{SA-TS}$ (%)
*chr*12a	9552	9552	23.96	9552	0.00	0.00
*chr*12b	9742	9742	23.74	9742	0.00	0.00
*chr*12c	11156	11156	23.76	11156	0.00	0.00
*chr*15a	9896	9896	56.92	9896	0.00	0.00
*chr*15b	7990	7990	56.82	7990	0.00	0.00
*chr*15c	9504	9504	58.09	9504	0.00	0.00
*chr*18a	11098	11098	119.66	11098	0.00	0.00
*chr*18b	1534	1534	119.29	1534	0.00	0.00
*chr*20a	2192	2224.9	179.20	2192	1.50	0.00
*chr*20b	2298	2306.7	181.59	2298	0.38	0.00
*chr*20c	14142	14142	177.30	14142	0.00	0.00
*chr*22a	6156	6181.3	257.49	6156	0.41	0.00
*chr*22b	6194	6265.2	257.48	6194	1.15	0.00
*chr*25a	3796	3811	427.97	3796	0.40	0.00

Tables 7.5 and 7.6 summarize the results of testing on *had* and *kra* problem instances. The average gaps of the solutions are less than 0.75%. For each problem instance, the hybrid algorithm is again able to obtain the best known/optimal solutions.

Table 7.5 Computational results of Algorithm SA-TS on problem class *had*

Benchmark problem	Optimal/Best known solution	Average Solution	Average CPU time (seconds)	Best Solution	$\Phi_{(1)}^{SA-TS}$ (%)	$\Phi_{(2)}^{SA-TS}$ (%)
*had*12	1652	1652	23.95	1652	0.00	0.00
*had*14	2724	2735	43.51	2724	0.40	0.00
*had*16	3720	3721	72.97	3720	0.03	0.00
*had*18	5358	5358	116.05	5358	0.00	0.00
*had*20	6922	6927.2	176.81	6922	0.08	0.00

Table 7.6 Computational results of Algorithm SA-TS on problem class *kra*

Benchmark problem	Optimal/Best known solution	Average Solution	Average CPU time (seconds)	Best Solution	$\Phi_{(1)}^{SA-TS}$ (%)	$\Phi_{(2)}^{SA-TS}$ (%)
*kra*30a	88900	89554.5	893.24	88900	0.74	0.00
*kra*30b	91420	91420	893.63	91420	0.00	0.00
*kra*32	88700	88700	1160.07	88700	0.00	0.00

Table 7.7 Computational results of Algorithm SA-TS on problem class *nug*

Benchmark problem	Optimal/Best known solution	Average Solution	Average CPU time (seconds)	Best Solution	$\Phi_{(1)}^{SA-TS}$ (%)	$\Phi_{(2)}^{SA-TS}$ (%)
*nug*12	578	578	23.66	578	0.00	0.00
*nug*14	1014	1014	43.11	1014	0.00	0.00
*nug*15	1150	1150	56.52	1150	0.00	0.00
*nug*20	2570	2570	177.39	2570	0.00	0.00
*nug*21	2438	2438	215.25	2438	0.00	0.00
*nug*22	3596	3596	258.28	3596	0.00	0.00
*nug*24	3488	3488	367.42	3488	0.00	0.00
*nug*25	3744	3744	430.63	3744	0.00	0.00
*nug*27	5234	5234	586.25	5234	0.00	0.00
*nug*28	5166	5166.9	675.75	5166	0.02	0.00
*nug*30	6124	6124.4	888.68	6124	0.01	0.00

Table 7.7 is a summary of the results for the *nug* type of benchmarks. The results indicate that these problem instances do not pose much difficulty for the proposed hybrid algorithm to obtain good solutions as the values of $\Phi_{(1)}^{SA-TS}$ are not more than 0.02%. All the best known/optimal solutions can be obtained within reasonable computation time.

For larger problem instances ($n \geq 20$), the proposed algorithm requires longer computation time. Nevertheless, it is able to find the best solutions for all problem instances. The optimal solution of the most difficult problem, namely the *nug*30, can be solved within 15 minutes.

The following tables show the results of testing on *rou*, *scr* and *sko* problem instances. The values of $\Phi_{(1)}^{SA-TS}$ are not more than 0.03% for *rou* and *scr* problem instances, while the maximum value of $\Phi_{(1)}^{SA-TS}$ is only 0.18% for *sko* problem instance. For *sko*49 and *sko*56, the values of $\Phi_{(2)}^{SA-TS}$ are around 0.1% from the optimal/best known solution. The longest CPU time required to obtain the solution is about 3 hours for *sko*56.

Table 7.8 Computational results of Algorithm SA-TS on problem class *rou*

Benchmark problem	Optimal/Best known solution	Average Solution	Average CPU time (seconds)	Best Solution	$\Phi_{(1)}^{SA-TS}$ (%)	$\Phi_{(2)}^{SA-TS}$ (%)
*rou*12	235528	235528	23.75	235528	0.00	0.00
*rou*15	354210	354210	56.61	354210	0.00	0.00
*rou*20	725522	725742.7	176.76	725522	0.03	0.00

Table 7.9 Computational results of Algorithm SA-TS on problem class *scr*

Benchmark problem	Optimal/Best known solution	Average Solution	Average CPU time (seconds)	Best Solution	$\Phi_{(1)}^{SA-TS}$ (%)	$\Phi_{(2)}^{SA-TS}$ (%)
*scr*12	31410	31410	23.73	31410	0.00	0.00
*scr*15	51140	51140	56.38	51140	0.00	0.00
*scr*20	110030	110030	176.29	110030	0.00	0.00

Table 7.10 Computational results of Algorithm SA-TS on problem class *sko*

Benchmark problem	Optimal/Best known solution	Average Solution	Average CPU time (seconds)	Best Solution	$\Phi_{(1)}^{SA-TS}$ (%)	$\Phi_{(2)}^{SA-TS}$ (%)
*sko*42	15812	15833.8	5699.98	15812	0.14	0.00
*sko*49	23386	23424.5	6353.46	23410	0.16	0.10
*sko*56	34458	34520.4	10867.03	34494	0.18	0.10

Table 7.11 Computational results of Algorithm SA-TS on problem class *tai*

Benchmark problem	Optimal/Best known solution	Average Solution	Average CPU time (seconds)	Best Solution	$\Phi_{(1)}^{SA-TS}$ (%)	$\Phi_{(2)}^{SA-TS}$ (%)
*tai*10a	135028	135028	23.99	135028	0.00	0.00
*tai*12a	224416	224416	46.86	224416	0.00	0.00
*tai*15a	388214	388214	101.71	388214	0.00	0.00
*tai*17a	491812	491812	156.03	491812	0.00	0.00
*tai*20a	703482	704610.2	374.43	703482	0.16	0.00
*tai*25a	1167256	1182462.3	1009.34	1175490	1.30	0.71
*tai*30a	1818146	1845611.7	2138.80	1833020	1.51	0.82
*tai*35a	2422002	2484348.1	3127.62	2477054	2.57	2.27
*tai*40a	3139370	3228315.1	5341.07	3207852	2.83	2.18
*tai*50a	4938796	5122386.6	12645.33	5115612	3.72	3.58
*tai*60a	7205962	7463484.2	13440.62	7417240	3.57	2.93
*tai*80a	13515450	13997867.4	15644.29	13938662	3.57	3.13
*tai*100a	21054656	21788679.9	22109.45	21689698	3.49	3.02

Table 7.12 Computational results of Algorithm SA-TS on problem class *wil*

Benchmark problem	Optimal/Best known solution	Average Solution	Average CPU time (seconds)	Best Solution	$\Phi_{(1)}^{SA-TS}$ (%)	$\Phi_{(2)}^{SA-TS}$ (%)
*wil*50	48816	48867.7	11570.09	48850	0.11	0.07
*wil*100	273038	273406.3	21982.86	273240	0.13	0.07

As shown in Tables 7.11 and 7.12, for the *tai* and *wil* type benchmarks, the performance of the SA–TS is still acceptable with values of $\Phi_{(1)}^{SA-TS}$ and $\Phi_{(2)}^{SA-TS}$ are not more than 3.72% and 3.58%, respectively. For larger problems (with $n > 20$), the best known/optimal solutions

cannot be found. It is likely that with greater number of iterations, the outcome may improve with possibility of obtaining the best known/optimal solutions for some of the benchmarks.

In summary, we show that QAP can be used to formulate the examination timetabling problem since both problems have similar characteristics. We propose a new hybrid algorithm for some QAP benchmark problems. The algorithm incorporates the advantages of three different algorithms: GRASP, Simulated Annealing and Tabu Search algorithms. It appears to be an efficient algorithm for the QAP. For some standard benchmark problems, this algorithm is able to obtain the optimal or the best known solutions. The computation time required is reasonable especially for problem instances with modest size.

7.3 The Extended Examination Timetabling Problem

Quadratic Semi-Assignment Problem (QSAP) is a special case used to model clustering and partitioning problems (Hansen and Lih, 1992). In QSAP, we have n facilities and m locations where $n > m$. All facilities have to be allocated to m locations so as to minimize the overall distance covered by the flow of materials moving between different facilities. Again, QSAP can be used to formulate another extension of the basic examination timetabling problem that would be explained below.

In the basic examination timetabling model (QAP model), the first order conflict cannot occur because one time period can only be scheduled for one examination. However, in many real world problems, the number of resources (rooms) available can be more than necessary and the number of examinations is greater than the number of time periods available. In this situation, it is possible to assign more than one examination to a time period. Thus, constraint (7.3) can

be relaxed and the examination problem can then be formulated as a Quadratic Semi-Assignment Problem (QSAP).

The details of the mathematical model are presented in the following sub-section.

7.3.1 The Mathematical Programming Model

The following models assume that each time period can accommodate more than one examination. Let Cap_v be number of classrooms available in time period $v\left(1 \le v \le n\right)$. Since each time period can be scheduled for more than one examination, the first order conflict might occur. In order to avoid this situation, we introduce different values of c'_{vw}:

$$c'_{vw} = \begin{cases} 1/(d_{vw})^{\eta} & |v-w| > 1 \\ \mu & |v-w| = 1 \\ \hat{M} & v = w \end{cases} \tag{7.10}$$

where

$\hat{M}$ = a very large number

η = a marginal product of time period

μ = a number greater than 1, and is typically set to 10

Here, we investigated cost function that is inversely proportional to distance with the power order η. The parameter η emphasizes the importance level of the conflict. Setting η = 0 means that we are only concerned about the first and second order conflicts and treat higher order conflicts as equally important. For example, there is no difference in terms of c'_{vw} value if the "distance" between two time periods *v* and *w*, |*v* – *w*|, is either two periods or greater than two periods. On the other hand, the cost between two time periods is diminishing for η = 1. A higher value of d_{vw} will result in a lower value of c'_{vw}. In this case, we concern about higher order conflicts.

[EP Model II]

Minimize $$Z_{EP\_II} = \sum_{t=1}^{n}\sum_{u=1}^{n}\sum_{v=1}^{n}\sum_{w=1}^{n} f'_{tu}c'_{vw}x_{tv}x_{uw} \tag{7.11}$$

subject to:

$$\sum_{t=1}^{n} x_{tv} \le Cap_v \quad (1 \le v \le n) \tag{7.12}$$

Constraints (7.3), (7.4)

EP Model II still assumes that the number of courses is equal to the number of time periods. Constraint (7.12) ensures that the maximum number of examinations scheduled at any time period is Cap_v. In this study, it is assumed that $Cap_v = Cap$ $(1 \le v \le n)$. The formulation of QSAP can always be adjusted to specific requirement of a certain university as stated in Bullnheimer (1998).

To further relax the restriction that the number of examinations and the number of time periods are equal, the more general examination scheduling problem with n slots or time periods and e ($e \ge n$) examinations to be scheduled is being considered in EP Model III.

[EP Model III]

Minimize $$Z_{EP\_III} = \sum_{t=1}^{e}\sum_{u=1}^{e}\sum_{v=1}^{n}\sum_{w=1}^{n} f'_{tu}c'_{vw}x_{tv}x_{uw} \tag{7.13}$$

subject to:

$$\sum_{t=1}^{e} x_{tv} \le Cap_v \quad (1 \le v \le n) \tag{7.14}$$

$$\sum_{v=1}^{n} x_{tv} = 1 \quad (1 \le t \le e) \tag{7.15}$$

$$x_{tv} \in \{0,1\} \quad (1 \le t \le e, 1 \le v \le n) \tag{7.16}$$

The quadratic form of QSAP can always be linearized by introducing new variables $y_{tvuw} = x_{tv}x_{uw}$ (Burkard, 1990) and constraints (7.18) and (7.19). Therefore, we can write EP Model III as the following model, EP Model IV.

[EP Model IV]

Minimize $$Z_{EP\_IV} = \sum_{t=1}^{e}\sum_{u=1}^{e}\sum_{v=1}^{n}\sum_{w=1}^{n} f'_{tu}c'_{vw}y_{tvuw} \tag{7.17}$$

subject to:

Constraints (7.14), (7.15), (7.16)

$$\sum_{u=1}^{e} y_{tvuw} \le Cap_v x_{tv} \qquad (1 \le t \le e, 1 \le v, w \le n) \tag{7.18}$$

$$\sum_{w=1}^{n} y_{tvuw} = x_{tv} \qquad (1 \le t, u \le e, 1 \le v \le n) \tag{7.19}$$

$$y_{tvuw} \in \{0,1\} \qquad (1 \le t, u \le e, 1 \le v, w \le n) \tag{7.20}$$

Since $y_{tvuw} = x_{tv}x_{uw} = x_{uw}x_{tv} = y_{uwtv}$, an additional constraint (7.26) can be included in EP Model IV. Finally, the entire model can be represented as EP Model V.

[EP Model V]

Minimize $$Z_{EP\_V} = \sum_{t=1}^{e}\sum_{u=1}^{e}\sum_{v=1}^{n}\sum_{w=1}^{n} f'_{tu}c'_{vw}y_{tvuw} \tag{7.21}$$

subject to:

$$\sum_{t=1}^{e} x_{tv} \le Cap_v \qquad (1 \le v \le n) \tag{7.22}$$

$$\sum_{v=1}^{n} x_{tv} = 1 \qquad (1 \le t \le e) \tag{7.23}$$

$$\sum_{u=1}^{e} y_{tvuw} \le Cap_v x_{tv} \qquad (1 \le t \le e, 1 \le v, w \le n) \tag{7.24}$$

$$\sum_{w=1}^{n} y_{tvuw} = x_{tv} \qquad (1 \le t, u \le e, 1 \le v \le n) \tag{7.25}$$

$$y_{tvuw} = y_{uwtv} \qquad (1 \le t,u \le e, 1 \le v,w \le n) \tag{7.26}$$

$$x_{tv} \in \{0,1\} \qquad (1 \le t \le e, 1 \le v \le n) \tag{7.27}$$

$$y_{tvuw} \in \{0,1\} \qquad (1 \le t,u \le e, 1 \le v,w \le n) \tag{7.28}$$

7.3.2 The Lower Bound

Two problems, EP Model IV and EP Model V, can be relaxed by replacing the integrality constraint $x_{tv} \in \{0,1\}$ to $x_{tv} \ge 0$ and $y_{tvuw} \in \{0,1\}$ to $y_{tvuw} \ge 0$. The relaxed problems are represented as EP Model VI and EP Model VII, respectively. EP Model VI is treated as the lower bound of EP Model IV, while EP Model VII is treated as the lower bound of EP Model V, respectively. Let Z_{EP_VI} be the objective function value of EP Model VI and Z_{EP_VII} be the objective function value of EP Model VII.

Preposition 7.1. $\boldsymbol{Z_{EP\_VI} = LB \le Z_{EP\_V}}$

PROOF. Suppose the integrality constraints $x_{tv} \in \{0,1\}, y_{tvuw} \in \{0,1\}$ in EP Model IV are replaced by $x_{tv} \ge 0, y_{tvuw} \ge 0$ to generate the relaxed problem EP Model VI. Every feasible solution to EP Model V is always a feasible solution to EP Model VI. If the optimal solution x^* of EP Model V does exist, it can be easily proven that $Z_{EP\_VI} = Z_{EP\_V}$.

Lemma 7.1. $\boldsymbol{Z_{EP\_V}}$ **= OPTIMAL**

PROOF. This lemma is adapted from Drezner (1995). We show that $y_{tvuw} = x_{tv}x_{uw}$. Two different cases are distinguished:

Case 1: Suppose $y_{tvuw} = 1$, the value of $x_{tv} = 1$ [by equation (7.24)] and $y_{uwtv} = 1$ [by equation (7.26)], thus $x_{uw} = 1$ as well as $x_{uw}x_{tv} = y_{uwtv}$.

Case 2: Suppose $y_{tvuw} = 0$, by a contradiction, we assume that $x_{tv} = x_{uw} = 1$. Since $x_{tv} = 1$, we get $x_{ta} = 0$ for all $a \neq v$ [by equation (7.23)]. Refer to equation (7.24), $y_{tauw} = 0$ for all $a \neq v$. By equation (7.26), $y_{uwta} = y_{tauw} = 0$. Since $x_{uw} = 1$, $y_{uwtv} = 1$ by equation (7.25) because all the other terms in the sum are equal to zero. This constitutes a contradiction.

Lemma 7.2. $\boldsymbol{Z_{EP\_VI} \leq Z_{EP\_IV}}$

PROOF: This lemma can be proven by checking that every feasible solution to EP Model IV is always a feasible solution to EP Model VI. If the solution of EP Model IV is integer, it can be easily proven that $Z_{EP\_VI} = Z_{EP\_IV}$.

Lemma 7.3. $\boldsymbol{Z_{EP\_VII} \leq Z_{EP\_V}}$

PROOF: Similar to lemma 7.2, this lemma can be proven by checking that every feasible solution to EP Model V is always a feasible solution to EP Model VII.

Lemma 7.4. $\boldsymbol{Z_{EP\_IV} \leq Z_{EP\_V}}$

PROOF: Since EP Model V is an extension of EP Model IV by adding constraint (7.26), every feasible solution to EP Model V is always a feasible solution to EP Model IV.

Lemma 7.5. $\boldsymbol{Z_{EP\_VI} \leq Z_{EP\_VII}}$

PROOF: Similar to lemma 7.4, by checking that every feasible solution to EP Model VII is a feasible solution to EP Model VI.

Preposition 7.2. $\boldsymbol{LB \leq Z_{EP\_VII} \leq OPTIMAL}$

PROOF: Let Z_{EP_VI} = LB and by lemma 7.5 $Z_{EP\_VI} \leq Z_{EP\_VII}$, we can conclude that LB $\leq Z_{EP\_VII}$. Let Z_{EP_V} = OPTIMAL (lemma 7.1) and $Z_{EP\_VII} \leq Z_{EP\_V}$ (lemma 7.3), we get $Z_{EP\_VII} \leq$ OPTIMAL.

7.3.3 The Proposed Algorithm

In the previous chapter, we have introduced a new hybrid metaheuristic for QAP. This algorithm is able to provide very good results. In order to solve the examination problem, we adopt the algorithm with some modifications. The entire algorithm comprises of two main phases: (1) construction, and (2) improvement. The GRASP is used to initialize a solution in the first phase, while a combination of Simulated Annealing and Tabu Search algorithms is for improving the solution in the second phase. Each phase is presented and described in detail below.

7.3.3.1 The Construction Phase

In the previous chapter, part of GRASP was implemented to build an initial solution for the QAP. Due to some of the general requirements of the examination timetabling problem differing from those of QAP, such as each time period can accommodate more than one examination and there is a limit of the number of examinations per time period, we need to modify GRASP presented in Section 7.2 accordingly. The details of the construction process of GRASP are described in Figure 7.7.

Construction Phase ( )

(1) Sort the $(e^2-e)/2$ flow entries in *FLOW* in increasing order, such that $f'_{t_1u_1} \geq f'_{t_2u_2} \geq \ldots \geq f'_{t_{\lfloor (e^2-e)/2 \rfloor} u_{\lfloor (e^2-e)/2 \rfloor}}$.

(2) Sort the $(n^2-n)/2$ cost entries in *COST* in non-increasing order, such that $c'_{v_1w_1} \leq c'_{v_2w_2} \leq \ldots \leq c'_{v_{\lfloor (n^2-n)/2 \rfloor} w_{\lfloor (n^2-n)/2 \rfloor}}$.

(3) Calculate the cost interaction $f'_{t_1u_1}c'_{v_1w_1}, f'_{t_2u_2}c'_{v_2w_2}, \ldots, f'_{t_{\lfloor \beta((n^2-n)/2) \rfloor} u_{\lfloor \beta((n^2-n)/2) \rfloor}} c'_{v_{\lfloor \beta((n^2-n)/2) \rfloor} w_{\lfloor \beta((n^2-n)/2) \rfloor}}$, sort them in increasing order and keep the smallest $\lfloor \beta((n^2-n)/2) \rfloor$ elements as the *candidate list*, where β is the second candidate restriction parameter $(0 < \beta < 1)$.

(4) Select a couple of assignment pairs from the *candidate list* randomly.

(5) Calculate C_{tv}, the cost of assigning examination *t* to time period *v*, with respect to the already-made assignments, Γ and capacity of each time period, Cap_v:
$C_{tv} = \sum_{(u,w) \in \Gamma} f'_{tu}c'_{vw}$ where $\Gamma = \{(u_1,w_1),(u_1,w_1),\ldots,(u_{\tilde{r}},w_{\tilde{r}})\}$,

(6) Set *o* = the number of unassigned examinations and *m* = the number of unassigned examination-time period pairs

(7) Determine the $\lfloor \gamma m \rfloor$ examination-time period pairs having the smallest C_{tv} values.

(8) Select an examination-time period pair (t,v) randomly from the list generated in Step 7.

(9) Update the set $\Gamma = \Gamma \cup (t,v)$

(10) Set $o = o - 1$

(11) Repeat Steps 5 – 10 until *o* = 0

Figure 7.7 Construction phase of Modified GRASP algorithm

For illustration of the construction phase in the modified GRASP algorithm, we provide a numerical example as follows. Consider the extended examination scheduling problems with *n* (= 5) slots or time periods and *e* (= 6) examinations to be scheduled as well as two matrices *FLOW* = [f'_{tu}] and *COST* = [c_{vw}].

FLOW

	1	2	3	4	5	6
1	0	10	0	15	5	3
2		0	2	0	25	26
3			0	7	8	12
4				0	2	0
5					0	1
6						0

COST

	1	2	3	4	5
1	1000	10	0.5	0.33	0.25
2		1000	10	0.5	0.33
3			1000	10	0.5
4				1000	10
5					1000

Figure 7.8 A numerical example for illustrating modified GRASP algorithm

Let β = 0.5 and γ = 0.5, we sort 15 flow entries in *FLOW*, keeping the 5 largest values: $f'_{26} \geq f'_{25} \geq f'_{14} \geq f'_{36} \geq f'_{12}$ and sort 10 cost entries in *COST*, keeping the 5 smallest values: $c'_{15} \leq c'_{14} \leq c'_{25} \leq c'_{13} \leq c'_{24}$. Then, we calculate the costs of interaction $f'_{26}c'_{15}$, $f'_{25}c'_{14}$, $f'_{14}c'_{25}$, $f'_{36}c'_{13}$, $f'_{12}c'_{24}$ and keep the smallest 3 elements: $f'_{14}c'_{25} \leq f'_{12}c'_{24} \leq f'_{36}c'_{13}$ as the candidate list (Figure 7.9).

$f'_{14}c'_{25}$	$f'_{12}c'_{24}$	$f'_{36}c'_{13}$	$f'_{26}c'_{15}$	$f'_{25}c'_{14}$
5	5	6	6.5	8.33

Figure 7.9 A numerical example of the costs of interaction in modified GRASP algorithm

A couple of assignments having the smallest interaction cost from the candidate list randomly is then selected. Assuming that $f'_{12}c'_{24}$ is selected, meaning that courses 1 and 2 are scheduled on time periods 2 and 4, respectively, we continue to schedule unscheduled examinations to one time period at a time.

Let Γ = {(1,2), (2,4)} be the set of already-made assignment, we calculate C_{tv}, the cost of assigning examination t to time period v, with respect to the already-made assignment, Γ, and the capacity of time period v. We sort these values in increasing order and we keep the smallest 10 elements: C_{31}, C_{32}, C_{45}, C_{44}, C_{33}, C_{35}, C_{61}, C_{51}, C_{41}, C_{43} (Figure 7.10). We then select one element randomly. Assuming that C_{32} is selected, we then update the set $\Gamma = \Gamma \cup \{(3,2)\}$. Similar to examination 1, examination 3 can be scheduled on time period 2 since the capacity of time period 2 is still enough. This process in continued until all unscheduled examinations have been scheduled.

C_{31}	C_{33}	C_{34}	C_{44}	C_{51}	C_{41}	C_{43}	C_{53}	C_{54}
0.5	1	2	7.5	11.25	15	15	17.5	27.5

Figure 7.10 A numerical example of C_{tv} in modified GRASP algorithm

7.3.3.2 The Improvement Phase

Having an initial feasible solution, we turn our attention towards the improvement of that solution. In this study, we focus on the hybridization of heuristics in order to solve the examination timetabling problem. The algorithm applied in this phase to improve the solution is adapted from Algorithm SA-TS described in Section 7.2. As described in Chapter V, Algorithm SA-TS outperforms other proposed hybrid algorithms. Figure 7.11 summarizes the details of the proposed algorithm.

Algorithm SA-TS ( )
(1) Initialize the parameters
(2) Set the best solution, *best_sol* = *initial_sol*
(3) Set the current solution, *current_sol* = *initial_sol*
(4) Set the total number of iterations, *num_iter* = 0
(5) Set the total number of iterations without improvement, *no_improv* = 0
(6) **While** the total number of iterations, *num_iter* is less than the preset maximum number of iterations, *outer_loop* do:
(7) Repeat *inner_loop* times:
(8) Select an examination *i* randomly
(9) Apply a partial sequential neighborhood search
(10) Find the best permutation π' with the smallest value of $\Delta(\pi,\pi',t)$
(11) **If** $\Delta(\pi,\pi',t) < 0$
(12) Update the current solution, *current_sol*
(13) Update tabu list
(14) **If** *current_sol* is better than *best_sol*
(15) Update the best solution, *best _sol* = *current_sol*
(16) **Else**
(17) Choose a random number *r* uniformly from [0,1]
(18) *no_improv*:= *no_improv*+1
(19) Check whether the best permutation is tabu or not
(20) **If** $r < exp^{-\Delta(\pi,\pi',t)/T_{num\_iter}}$ and the new solution is not tabu
(21) Update the current solution, *current_sol*
(22) Update tabu list
(23) **Else**
(24) Return to the current solution, *current_sol*
(25) Update tabu list
(26) Update temperature $T_{num\_iter} := \alpha T_{num\_iter}$
(27) **If** (*no_improv* > *limit*)
(28) Apply the intensification strategy
(29) Set *no_improv*:= 0
(30) *num_iter* := *num_iter*+1
(31) **End while**
(32) Report the best solution, *best_sol*

Figure 7.11 Algorithm SA-TS for the extended examination timetabling problem

The neighborhood is defined by reallocating an examination of the current solution π to another different time period (single move) such that a better solution π' is derived. It is necessary to ensure that the maximum number of examinations scheduled at any time period Cap_v is not being exceeded. Instead of a random neighborhood search, a partial sequential neighborhood search is used, which involves examining all other potential moves sequentially with respect to time periods for an examination of the current solution π.

The objective function difference $\Delta(\pi,\pi',t)$ obtained by exchanging the time period of examination *t*, $\pi(t)$ and $\pi'(t)$, using the following equation:

$$\Delta(\pi,\pi',t) = 2\times\sum_{\substack{a=1\\a\neq t}}^{e} f'_{at}(c'_{\pi(a)\pi'(t)} - c'_{\pi(a)\pi(t)}) \tag{7.29}$$

The selected move is the one with the best $\Delta(\pi,\pi',t)$ value. The new permutation is then evaluated by the acceptance-rejection procedure in SA. We incorporate features from Tabu Search, such as the tabu length, tabu list and intensification strategy in the algorithm.

The tabu list contains examination-time period pairs that have been visited in the last *length* iterations. For a given iteration, if a pair $(t,\pi'(t))$ belongs to the tabu list, it is not allowed to accept the exchange of the time periods $\pi(t)$ and $\pi'(t)$, unless this exchange gives a strictly better objective function value (*aspiration level criteria*). At any temperature *T*, the neighborhood search is repeated until a certain number of iterations, *inner_loop*, has been performed.

7.3.4 Computational Results

The computational results of applying the proposed algorithm to solve several randomly generated problems are shown. Several data sets are generated in such a way that the data sets

correspond to three various parameters: the number of examinations, the number of time periods and the number of classrooms available per time period. These data are empirical data which are comparable to the real data used in the examination timetabling problem in an engineering faculty of a university in Indonesia.

We set the number of time periods to be 20. This is by assuming that one week consists of 5 working days, and each working day consists of 4 time periods, of which the length of each time period is 2 to 3 hours. The value of parameter η is set to 1. The characteristics of the problem instances used in the computational experiments are shown in Table 7.13. Each data set consists of the number of examinations that need to be scheduled and the total number of available time periods. For instance, data set 10×10 consists of 10 examinations and 10 time periods available.

Table 7.13 Characteristics of the extended examination timetabling problem data sets

No	Data set	Number of examinations *e*	Number of time periods *n*	Number of classrooms *Cap*
1	10×10×3	10	10	3
2	20×20×3	20	20	3
3	40×20×4	40	20	4
4	60×20×5	60	20	5
5	80×20×6	80	20	6
6	100×20×7	100	20	7

Table 7.14 summarizes the values of the parameters used in the computational study which are determined experimentally to ensure a compromise between the computation time and the solution quality.

Table 7.14 Parameter settings for Algorithm SA-TS
(the extended examination timetabling problem)

Parameter	Value
Maximum number of iterations, *outer_loop*	$50 \times e$
Initial temperature, T_0	1000
Number of neighborhood moves at each temperature T, *inner_loop*	$100 \times n$
Cooling factor, α	0.9
Number of non-improvement iterations prior to intensification, *limit*	$0.01 \times$ *outer_loop*
Length of tabu list, *length*	$e/2$

The algorithm is coded in C++ and executed on a 2.67 GHz Intel (R) Core™ 2 Duo CPU with 3 GB of RAM under the Microsoft Windows Vista Operating System. The algorithm is repeated for 20 runs, with the average objective function value, the best objective function value and the average computation time being tabulated. To see if the proposed algorithm is an improvement over a pure SA algorithm, we also applied a pure SA algorithm for all data sets.

Table 7.15 Computational results of Algorithm SA-TS

No	Data set	Algorithm SA			Algorithm SA-TS		
		Average objective function value	Best objective function value	Average CPU time (seconds)	Average objective function value	Best objective function value	Average CPU time (seconds)
1	10×10×3	14.8	14.8	1.29	14.8	14.8	1.31
2	20×20×3	123.36	121.68	18.64	121.84	121.68	20.26
3	40×20×4	642.24	628.48	65.61	634.82	627.66	68.55
4	60×20×5	2743.22	2657.32	140.46	2699.42	2652.08	140.39
5	80×20×6	7723.47	7410.62	236.98	7620.33	7281.68	246.20
6	100×20×7	20275.46	18167.34	340.90	19456.78	17583.36	350.52

In general, both algorithms perform very well and obtained quite good solutions within reasonable computation time (Table 7.15). It indicates that the performance of the hybrid algorithm, Algorithm SA-TS, is better than the pure SA in terms of the average and the best

objective function values obtained. The computation time needed by the hybrid algorithm also compares favourably with that of the pure SA.

Table 7.16 provides a numerical comparison between the lower bound, the average as well as best objective function values. The lower bounds Z_{EP_VII} were obtained by using ILOG OPL Studio 4.2 on the same processor and operating system used for implementing the proposed algorithms.

The heading $\Phi_{(1)}^{SA-TS}$ refers to the percentage deviation between the average objective function value of the solutions obtained by Algorithm SA-TS and the lower bound, while the heading $\Phi_{(2)}^{SA-TS}$ refers to the percentage deviation between the best objective function value of the solutions obtained by Algorithm SA-TS and the lower bound.

Table 7.16 Comparison of the algorithm results and the lower bounds

No	Data Set	Average objective function value	Best objective function value	Lower Bound	$\Phi_{(1)}^{SA-TS}$ (%)	$\Phi_{(2)}^{SA-TS}$ (%)
1	10×10×3	14.8	14.8	12.2	21.31	21.31
2	20×20×3	121.84	121.68	93.02	30.98	30.81
3	40×20×4	634.82	627.66	509.06	24.70	23.30
4	60×20×5	2699.42	2652.08	2133.46	26.53	24.31
5	80×20×6	7620.33	7281.68	6158.14	23.74	18.24
6	100×20×7	19456.78	17583.36	14243.32	36.60	23.45

In overall, the average and the best percentage deviations are less than 37% and 31%, respectively. Since the proposed lower bound is mainly based on the linear relaxation of the problem, an interesting open problem that can be considered for future research work would be to develop better quality lower bounds for this problem. For instance, a Lagrangian relaxation approach can be considered for calculating a better lower bound.

7.4 Conclusions

In this chapter, we present the basic examination timetabling problem as a QAP problem. In this basic model, we assume that the number of examinations is equal to the number of time periods. A new hybrid algorithm was introduced to solve the problem. The algorithm incorporates three different algorithms: GRASP, Simulated Annealing and Tabu Search algorithms. It appears to be an efficient heuristic algorithm for solving the problem. For some standard benchmark problems, this algorithm is able to obtain the optimal or the best known solutions within reasonable computation time.

In this study, we present another suitable model for the extended examination timetabling problem where the number of examinations is larger than the number of time periods. The problem is represented as a Quadratic Semi-Assignment Problem (QSAP). This model allows the number of examinations is more than the number of time periods available and each time period can accommodate more than one examination.

Another hybrid algorithm based on a combination of GRASP, Simulated Annealing and Tabu Search was proposed for solving this problem that cannot be easily solved by commercial software. It is shown that the proposed algorithm yields good solutions within reasonable computation time compared with those of pure Simulated Annealing.

Subjects of possible areas for future research should be the inclusion of other constraints and the development of better lower bounds. We can look into ways of improving the proposed algorithm to obtain better solutions especially when the problem size is very large.

CHAPTER VIII

CONCLUSIONS AND FUTURE RESEARCH WORK

In this chapter, we summarize and discuss the major findings and contributions of this study. Some suggestions and recommendations for future work will also be presented.

8.1 Major Findings and Contributions

The timetabling problem has been studied in a wide variety of application domains including educational timetabling, transportation timetabling, employee timetabling, healthcare timetabling and sport timetabling. In this study, we focus on the educational timetabling problem at the university level, known as the university timetabling problem. The following figure represents the framework of the problem studied.

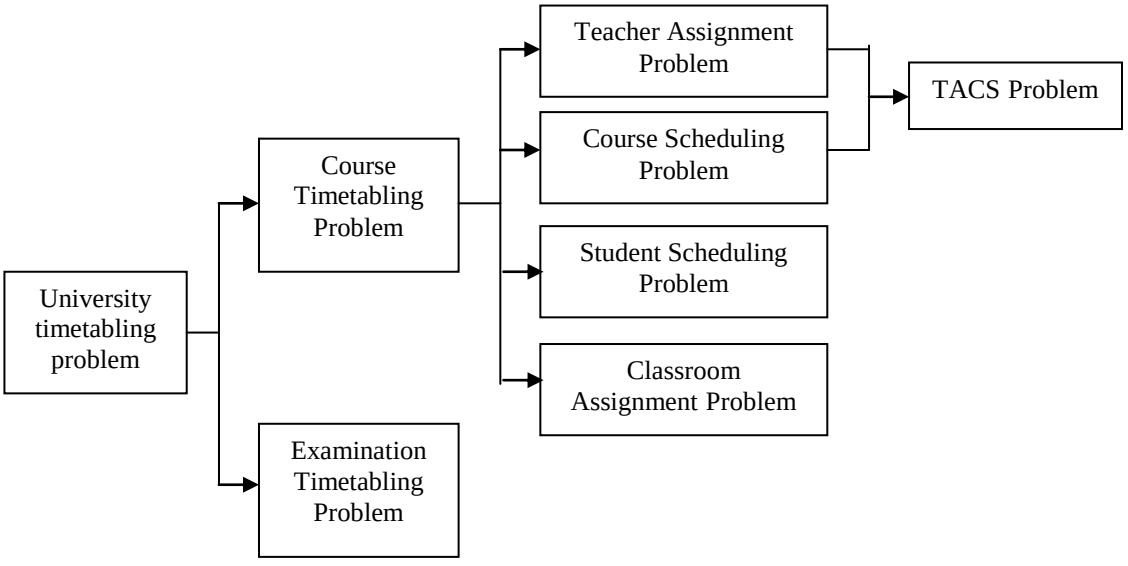


Figure 8.1 Framework of the study

The university timetabling problem can be classified into two categories: the course and examination timetabling problems. There are, of course, several differences between both categories. In the course timetabling problem, one course has to be scheduled into exactly one classroom. However, a number of examinations can often be scheduled into one room or an examination may be split across several classrooms in the examination timetabling problem.

In the university course timetabling problem, four different sub-problems are involved: course scheduling, student scheduling, teacher assignment and classroom assignment. In this study, we focus on two sub-problems: the teacher assignment and course scheduling problems. The problem of teacher assignment is how to assign teachers to courses while maximizing a preference function. In course scheduling problem, the main concern is how to schedule course sections to time periods over a week. It is often assumed that the allocation of teachers to courses has been done and fixed earlier before the actual scheduling of courses to time periods.

Motivated by the need to overcome this limitation of only considering one sub-problem, we studied a combination of teacher assignment and course scheduling simultaneously which is not commonly studied by other researchers. This combination is known as the Teacher Assignment – Course Scheduling problem (TACS problem).

An important contribution relating to the university course timetabling problem is on three different mathematical programming models proposed and presented in Chapter 3. These models, namely, TACS Model I, TACS Model II and TACS Model III represent a combination of the teacher assignment and the course scheduling problems. TACS Model I which is considered as the basic model only includes several common requirements of both sub-problems. TACS Model II includes some additional requirements such as courses can be divided into several sections and one course can be taught by more than one teacher. TACS

Model III which is the most complex model includes several other requirements which are comparable to those occurring in an engineering faculty of a university in Indonesia.

Computational experiments based on some randomly generated instances are summarized and discussed in detail in Chapter 3. Initially, the problems were solved by ILOG OPL Solver software. Unfortunately, some large-scale instances of TACS Model III could not be optimally solved.

In order to overcome the limitation of solving TACS Model III, two different methods, namely, a simple improvement heuristic and hybrid algorithms were developed in Chapters 4, 5 and 6. In Chapter 4, we introduced a simple improvement heuristic that applies the principles of a greedy heuristic. Since the results obtained by this heuristic are not good enough, we proposed four different hybrid algorithms in order to solve the problem.

In Chapter 5, three different hybrid algorithms, mainly based on a greedy heuristic, Simulated Annealing and Tabu Search, were developed. These algorithms are Algorithms SA1, SA2 and SA-TS. The computational experience shows that the proposed hybrid algorithms are able to produce good quality solutions for several random problem instances. It has been concluded that Algorithms SA-TS and SA2 outperform SA1 significantly in terms of objective function values obtained.

A possible reason is that the exploration of the neighborhood structure in SA1 is limited. The neighborhood consists of two operations: reallocation of teachers to courses and rescheduling courses to time periods. If the first operation is accepted, we continue to the second operation and evaluate whether these two operations are acceptable. Unfortunately, it might be possible that the teacher reallocation is rejected and the process is then terminated without proceeding

to the evaluation process. On the other hand, Algorithms SA2 and SA-TS applied the evaluation process at the end of each operation. Although the first operation is rejected, we still continue to the second operation by choosing another course that has to be scheduled to another set of day-time periods.

In Algorithm SA-TS, we also include several features from Tabu Search such as the aspiration level criterion and tabu list. We found that Algorithm SA-TS performs better than Algorithm SA2 does. It shows that unnecessary moves during high temperature can be avoided by including those features.

In Chapter 6, we have studied and presented a successful hybrid algorithm of the Lagrangian relaxation procedure and a modified Simulated Annealing (SA) algorithm, namely, Algorithm LR-SA, for solving the TACS problem. The performance of the proposed algorithm was measured by applying the algorithm to the same problems discussed in earlier chapters. We concluded that LR-SA outperforms the previous algorithm SA–TS in terms of solution quality. The Lagrangian relaxation approach is able to construct good quality initial solutions that would lead to better quality final solutions.

The idea of the hybrid algorithm that has been successfully applied in the course timetabling problem has motivated us to apply to another university timetabling problem, known as the examination timetabling problem. In the university examination timetabling problem, we notice that majority of the methods proposed were centered on the general concepts of graph theory or network analysis.

In this study, the basic examination timetabling problem was formulated as a Quadratic Assignment Problem (QAP). A new hybrid algorithm (Algorithm GRASP-SA-TS) that

combines several features of Greedy Randomized Adaptive Search Procedure (GRASP), Simulated Annealing and Tabu Search algorithms had been proposed in Chapter 7. In summary, the proposed algorithm was able to obtain the optimal or the best known solutions for several QAP benchmark problems within reasonable computation time.

Since the number of examinations can be greater than the number of time periods, the examination timetabling problem is reformulated as the Quadratic Semi-Assignment Problem (QSAP). The lower bound for the extended examination timetabling problem, which is based on the relaxation of the integrality constraints, was proposed. Algorithm GRASP-SA-TS was then modified in order to solve the problem. We conclude that the proposed algorithm yields good solutions within reasonable computation time compared with those of pure Simulated Annealing.

To sum up, the main contributions of this study are as follows:

- In the course timetabling context, two sub-problems: teacher assignment and course scheduling problem are discussed and solved simultaneously, which is not commonly studied by other researchers. This combination problem is known as TACS problem.
- Three different mathematical models are developed in order to represent the combination of teacher assignment and course scheduling problems simultaneously (TACS problem).
- Several hybrid algorithms based on hybridization of the Simulated Annealing and other methods are proposed in order to solve the TACS problem.
- The examination timetabling problem is formulated as QAP and QSAP. These formulations can represent first-order, second-order and higher order conflicts which cannot be captured in graph coloring formulation.
- The idea of the hybrid algorithm is extended and applied to solve the examination timetabling problem.

8.2 Limitations and Future Research Work

Different models of university timetabling problems and advancements of different new approaches have been proposed in this thesis. Major findings and contributions were summarized as above. Nevertheless, due to limitations involved in the current study, there are still some aspects that were not addressed yet and deserve further explorations.

Several randomly generated data sets of sizes that are comparable to those occurring in an engineering faculty of a university in Indonesia were proposed. We notice that one limitation of the research is the lack of real-world applications which can prove the robustness of the proposed approaches and methodologies.

As stated in Chapter 1, this thesis focuses on teacher assignment and course scheduling problems at the university level. The whole process of building a timetable consists of other problems: classroom assignment and student scheduling problems, which are not considered in this study. Similarly, this study did not cover the classroom assignment problem in the examination timetabling problem.

In this study, we focus on the course timetabling and examination timetabling problems which are comparable to those occurring in an engineering faculty of a university in Indonesia. Some other constraints that might be occurred in other universities are not considered, such as certain courses or examinations have to be scheduled consecutively, certain courses or examinations cannot be conducted at the same time period, the uniformity of workload and so on.

The following points of further research are based on the limitations in current thesis. From a broader perspective, there are some interesting research topics that can be recommended for future research.

Since one limitation of this research is the lack of real applications, more effort should be put on the incorporation of real case timetabling problems that would be considered as future research work. Each institution might have a different curriculum structure, additional requirements can be included in the models in order to make the models more realistic and acceptable, for instance, some courses or examinations cannot be scheduled together, some courses or examinations have to be conducted consecutively, the uniformity of the workload and so on.

A lot of variables have been used especially in TACS Model III, and hence ILOG OPL Studio software is not effective in solving the problem. There might be a better way to model the problem especially for the one is used to model the consecutive constraints. One possible way to consider in future research is to use column generation concept. Column generation can be beneficial for a large-scale problem with many variables but relatively few constraints (Desaulniers et al., 2005).

Another possible area for future research is to incorporate other sub-problems, such as classroom assignment and student scheduling into the process. This would transform the entire process of building course and examination timetables into a more complex and complete process. It would be beneficial to extend the problem into the whole process of building a complete timetable by considering other sub-problems as well.

There are several differences between school and university timetabling problems. In school timetabling problem, we often work with predefined classes and have few programs so that the

main problem is to determine teachers for given subjects and classes, and to construct appropriate teaching schedules. In universities, more programs are offered and faculty members/teachers may only teach few hours a week. Due to these differences, the study of building either a course or an examination timetable at the school level can be considered as future research work.

Another possible extension is to improve the proposed methods in order to obtain better solutions. For example, in the proposed algorithms, the Tabu Search framework was designed with only a short term memory (tabu list) and the intensification strategy. It might be useful to implement other strategies, such as diversification strategy. The diversification strategy encourages the search process to examine unvisited regions and to generate solutions that differ in various significant ways from those seen before. One possible diversification technique is constraint relaxation. Constraint relaxation is implemented by dropping selected constraints from the search space definition and adding to the objective weighted penalties for constraint violations.

REFERENCES

Abramson, D. Constructing School Timetables using Simulated Annealing: Sequential and Parallel Algorithms, Management Science, 37(1), pp. 98-113. 1991.

Abramson, D., Krishnamoorthy, M. and Dang, H. Simulated Annealing Cooling Schedules for the School Timetabling Problem, Asia-Pacific Journal of Operational Research, 16(1), pp. 1-22. 1999.

Adam, W.P. and Johnson, T.A. Improved Linear Programming-Based Lower Bounds for the Quadratic Assignment Problem. In Quadratic Assignment and Related Problems: DIMAS Workshop, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 16, ed by P.M. Pardalos and H. Wolkowicz, pp. 43-75. 1994.

Ahuja, R.K., Ergun, O., Orlin, J.B. and Punnen, A.P. A Survey of Very Large-Scale Neighborhood Search Techniques, Discrete Applied Mathematics, 123(1-3), pp. 75-102. 2002.

Ahuja, R.K., Orlin, J.B. and Tiwari, A. A Greedy Genetic Algorithm for the Quadratic Assignment Problem, Computers and Operations Research, 27, pp. 917-934. 2000.

Al-Yakoob, S.M. and Sherali, H.D. A Mixed Integer Programming Approach to a Class Timetabling Problem: A Case Study with Gender Policies and Traffic Considerations, European Journal of Operational Research, 180, pp. 1028-1044. 2007.

Al-Yakoob, S.M. and Sherali, H.D. Mathematical Programming Models and Algorithms for a Class-Faculty Assignment Problem, European Journal of Operational Research, 173, pp. 488-507. 2006.

Anderson, E.J. Theory and Methodology: Mechanisms for Local Search, European Journal of Operational Research, 88, pp. 139-151. 1996.

Andrew, G.M. and Collins, R. Matching Faculty to Courses, College and University, 46 (2), pp. 83-89. 1971.

Anstreicher, K.M. Recent Advances in the Solution of Quadratic Assignment Problems, Mathematical programming, 97 (1-2), pp. 27-42. 2003.

Applegate, D., Bixby, R., Chvátal, V. and Cook, W. On the Solution of the Traveling Salesman Problem, Documenta Mathematica, Extra Volume III, pp. 645-656. 1998.

Arani, T. and Lotfi, V. A Three Phased Approach to Final Exam Scheduling, IIE Transactions, 21 (1), pp. 86-95. 1989.

Arkin, E.M., Hassin, R. and Sviridenko, M. Approximating the Maximum Quadratic Assignment Problem, Information Processing Letters, 77 (1), pp. 13-16. 2001.

Asmuni, H., Burke, E.K., Garibaldi, J. and McCollum, B. Fuzzy Multiple Ordering Criteria for Examination Timetabling. In Practice and Theory of Automated Timetabling: 5th International Conference, Lecture Notes in Computer Science, Vol. 3616, ed by E.K. Burke and M. Trick, pp. 334-353. New York: Springer. 2005.

Asratian, A.S. and de Werra, D. A Generalized Class-Teacher Model for Some Timetabling Problems, European Journal of Operational Research, 143, pp. 531-542. 2002.

Aubin, J. and Ferland, J.A. A Large Scale Timetabling Problem, Computers and Operations Research, 16 (1), pp. 67-77. 1989.

Badri, M.A. A Two-Stage Multiobjective Scheduling Model for [Faculty-Course-Time] Assignments, European Journal of Operational Research, 94 (1), pp. 16-28. 1996.

Badri, M.A., Davis, D.L., Davis, D.F. and Hollingsworth, J. A Multi-Objective Course Scheduling Model: Combining Faculty Preferences for Courses and Times, Computers and Operations Research, 25 (4), pp. 306-316. 1998.

Balakrishnan, N., Lucena, A. and Wong, R.T. Scheduling Examinations to Reduce Second-Order Conflicts, Computers and Operations Research, 19, pp. 353-361. 1992.

Barham, A.M. and Westwood, J.B. A Simple Heuristic to Facilitate Course Timetabling, The Journal of the Operational Research Society, 29 (11), pp. 1055–1060. 1978.

Bellanti, F., Carello, G., Croce, F.D. and Tadei, R. A Greedy-Based Neighborhood Search Approach to a Nurse Rostering Problem, European Journal of Operational Research, 153 (1), pp. 28-40. 2004.

Benjaafar, S. Modeling and Analysis of Congestion in the Design of Facility Layouts, Management Science, 48 (5), pp. 679-704. 2002.

Bianchi, L., Birattari, M., Chiarandini, M., Manfrin, M., Mastrolilli, M., Paquete, L., Rossi-Doria, O. and Schiavinotto, T. Hybrid Metaheuristics for the Vehicle Routing Problem with Stochastic Demands. Technical Report No. IDSIA-06-05, Dalle Molle Institute for Artificial Intelligence. 2005.

Birbas, T., Daskalaki, S. and Housos, E. Timetabling for Greek High Schools, The Journal of the Operational Research Society, 48 (12), pp. 1191-1200. 1997.

Blum, C. and Roli, A. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison, ACM Computing Surveys, 35 (3), pp. 268-308. 2003.

Boufflet, J.P. and Negre, S. Three Methods used to Solve an Examination Timetabling Problem. In Practice and Theory of Automated Timetabling: First International Conference, Lecture Notes in Computer Science, Vol. 1153, ed by E.K. Burke and P. Ross, pp. 327-344. New York: Springer. 1996.

Bräannlund, U., Lindberg, P.O., Nõu, A. and Nilsson, J.E. Railway Timetabling using Lagrangian Relaxation, Transportation Science, 32 (4), pp. 358-369. 1998.

Breslaw, J.A. A Linear Programming Solution to the Faculty Assignment Problem, Socio - Economic Planning Sciences, 10, pp. 227-230. 1976.

Büdenbender, K., Grünert, T. and Sebastian, H.J. A Hybrid Tabu Search/Branch-and-Bound Algorithm for the Direct Flight Network Design Problem, Transportation Science, 34 (4), pp. 364-380. 2000.

Bullnheimer, B. An Examination Scheduling Model to Maximize Students' Study Time. In Practice and Theory of Automated Timetabling: Second International Conference, Lecture

Notes in Computer Science, Vol. 1408, ed by E.K. Burke and M.W. Carter, pp. 78-91. New York: Springer. 1998.

Burkard, R.E. and Bonniger, T. A Heuristic for Quadratic Boolean Programs with Applications to Quadratic Assignment Problems, European Journal of Operational Research, 17 (2), pp. 374-386. 1983.

Burkard, R.E. and Rendl, F. A Thermodynamically Motivated Simulated Simulation Procedure for Combinatorial Optimization Problems, European Journal of Operational Research, 17 (2), pp. 169-174. 1984.

Burkard, R.E. Locations with Spatial Interactions: the Quadratic Assignment Problem. In Discrete Location Theory, ed by P.B. Mirchandani and R.L. Francis, pp. 387-437. New York: John Wiley and Sons. 1990.

Burkard, R.E. Quadratic Assignment Problems, European Journal of Operational Research, 15 (3), pp. 283-289. 1984.

Burkard, R.E., Karisch, S.E. and Rendl, F. QAPLIB – A Quadratic Assignment Problem Library, Journal of Global Optimization, 10, pp. 391-403. 1997.

Burke, E.K and Newall, J.P. A Multistage Evolutionary Algorithm for the Timetable Problem, IEEE Transactions on Evolutionary Computation, 3 (1), pp. 63-74. 1999.

Burke, E.K. and Petrovic, S. Recent Research Directions in Automated Timetabling, European Journal of Operational Research, 140, pp. 266-280. 2002.

Burke, E.K., Bykov, Y. and Petrovic, S. A Multicriteria Approach to Examination Timetabling. In Practice and Theory of Automated Timetabling: 3rd International Conference, Lecture Notes in Computer Science, Vol. 2079, ed by E. Burke and W. Erben, pp. 118-131. New York: Springer. 2001.

Burke, E.K., Bykov, Y., Newall, J.P. and Petrovic, S. A Time-Predefined Local Search Approach to Exam Timetabling Problems, IIE Transactions 36 (6), pp. 509 – 528. 2004.

Burke, E.K., Cowling, P.I. and Keuthen, R. Effective Local and Guided Variable Neighborhood Search Methods for the Asymmetric Travelling Salesman Problem. In Applications of Evolutionary Computing: Evo Workshops 2001, Lecture Notes in Computer Science, Vol. 2037, ed by. E.J.W. Boers et al., pp. 203-212, New York: Springer. 2001.

Burke, E.K., de Causmaecker, P., Berghe, G.V. and Landeghem, H.V. The State of the Art of Nurse Rostering, Journal of Scheduling, 7 (6), pp. 441-499. 2004.

Burke, E.K., Elliman, D.G. Ford, P.H. and Weare, R.F. Examination Timetabling in British Universities: a Survey. In Practice and Theory of Automated Timetabling: First International Conference, Lecture Notes in Computer Science, Vol. 1153, ed by E.K. Burke and P. Ross, pp. 76-90. New York: Springer. 1996.

Burke, E.K., Kingston, J.H. and de Werra, D. Applications to Timetabling. In The Handbook of Graph Theory, ed by J. Gross and J. Yellen, pp. 445-474. Boca Raton: CRC Press. 2004.

Burke, E.K., Newal, J.P. and Weare, R.F. A Memetic Algorithm for University Exam Timetabling. In Practice and Theory of Automated Timetabling: First International Conference, Lecture Notes in Computer Science, Vol. 1153, ed by E.K. Burke and P. Ross, pp. 241-250. New York: Springer. 1996.

Burke, E.K., Newall, J.P. and Weare, R.F. A Simple Heuristically Guided Search for the Timetable Problem. In Proc. International ICSC Symposium on Engineering of Intelligent Systems, February 1998, Tenerife, Spain, pp. 574-579.

Carrasco, M.P. and Pato, M.V. A Multiobjective Genetic Algorithm for the Class/Teacher Timetabling Problem. In Practice and Theory of Automated Timetabling: 3rd International Conference, Lecture Notes in Computer Science, Vol. 2079, ed by E. Burke and W. Erben, pp. 3-17. New York: Springer. 2001.

Carter, M.W. A Comprehensive Course Timetabling and Student Scheduling System at the University of Waterloo. In Practice and Theory of Automated Timetabling: 3rd International Conference, Lecture Notes in Computer Science, Vol. 2079, ed by E. Burke and W. Erben, pp. 64-82. New York: Springer. 2001.

Carter, M.W. A Lagrangian Relaxation Approach to the Classroom Assignment Problem, INFOR, 27(2), pp. 230-244. 1989.

Carter, M.W. A Survey of Practical Applications of Examination Timetabling Algorithms, Operations Research, 34 (2), pp. 193-202. 1986.

Carter, M.W. and Laporte, G. Recent Developments in Practical Course Timetabling. In Practice and Theory of Automated Timetabling: Second International Conference, Lecture Notes in Computer Science, Vol. 1408, ed by E. Burke and M.W. Carter, pp. 3-19. New York: Springer. 1998.

Carter, M.W. and Laporte, G. Recent Developments in Practical Examination Timetabling. In Practice and Theory of Automated Timetabling: First International Conference, Lecture Notes in Computer Science, Vol. 1153, ed by E.K. Burke and P. Ross, pp. 3-21. New York: Springer. 1996.

Carter, M.W. and Tovey, C.A. When is the Classroom Assignment Problem Hard?, Operations Research, 40, Supp. No. 1, pp. S28-S39. 1992.

Carter, M.W., Laporte, G. and Lee, S.Y. Examination Timetabling: Algorithmic Strategies and Applications, The Journal of the Operational Research Society, 47 (3), pp. 373-383. 1996.

Cerny, V. A Thermodynamical Approach to the Travelling Salesman Problem: an Efficient Simulation Algorithm, Journal of Optimization Theory and Applications, 45, pp. 41-51. 1985.

Chalal, N. and de Werra, D. An Interactive System for Constructing Timetables on a PC, European Journal of Operational Research, 40, pp. 32 – 37. 1989.

Chiarandini, M. and Stützle, T. A Landscape Analysis for a Hybrid approximate Algorithm on a Timetabling Problem. Technical Report AIDA–03–05, TU Darmstadt. 2003.

Chiarandini, M., Birattari, M., Socha, K. and Rossi-Doria, O. An Effective Hybrid Algorithm for University Course Timetabling, Journal of Scheduling, 9 (5), pp. 403-432. 2006.

Chien, T.W, Balakrishnan, A. and Wong, R.T. An Integrated Inventory Allocation and Vehicle Routing Problem, Transportation Science, 23 (2), pp. 67-76. 1989.

Clements, D., Crawford, J., Joslin, D., Nemhauser, G., Puttlitz, M. and Savelsbergh, M. Heuristic Optimization: A Hybrid AI/OR Approach. In Proc. 3rd International Conference on Principles and Practice of Constraint Programming, November 1997, Schloss Hagenberg, Austria, pp. 1-12.

Connolly, D.T. An Improved Annealing Scheme for the QAP, European Journal of Operational Research, 46, pp. 93-100. 1990.

Cooper, T. and Kingston, J. The Complexity of Timetable Construction Problems. In Practice and Theory of Automated Timetabling: First International Conference, Lecture Notes in Computer Science, Vol. 1153, ed by E. Burke and P. Ross, pp. 283-295. New York: Springer. 1996.

Costa, D. A Tabu Search Algorithm for Computing an Operational Timetable, European Journal of Operational Research, 76, pp. 98-110. 1994.

Daskalaki, S. and Birbas, T. Efficient Solutions for a University Timetabling Problem through Integer Programming, European Journal of Operational Research, 160 (1), pp. 106-120. 2005.

Daskalaki, S., Birbas, T. and Housos, E. An Integer Programming Formulation for a Case Study in University Timetabling, European Journal of Operational Research, 153 (1), pp. 117-135. 2004.

David, P. A Constraint-Based Approach for Examination Timetabling using Local Repair Techniques. In Practice and Theory of Automated Timetabling: Second International Conference, Lecture Notes in Computer Science, Vol. 1408, ed by E.K. Burke and M.W. Carter, pp. 169-186. New York: Springer. 1998.

de Werra, D. An Introduction to Timetabling, European Journal of Operational Research, 19, pp. 151-162. 1985.

de Werra, D. Construction of School Timetables by Flow Methods, INFOR, 9 (1), pp. 12-22. 1971.

de Werra, D. The Combinatorics of Timetabling, European Journal of Operational Research, 96, pp. 504-513. 1997.

Desaulniers, G., Desrosiers, J. and Solomon, M.M. Column Generation. New York: Springer. 2005.

Dickey, J.W. and Hopkins, J.W. Campus Building Arrangement using Topaz, Transportation Research, 6, pp. 59-68. 1972.

Dimopoulou, M. and Miliotis, P. Implementation of A University Course and Examination Timetabling System, European Journal of Operational Research, 130, pp. 202-213. 2001.

Dorigo, M., Maniezzo, V., and Colorni, A. The Ant System: Optimization by a Colony of Cooperating Agents, IEEE Transaction on Systems, Man, and Cybernetics-Part B, 26 (2), pp. 29-41. 1996.

Dowsland, K.A. and Thompson, J. Ant Colony Optimization for the Examination Scheduling Problem, Journal of Operational Research Society, 56, pp. 426-438. 2005.

Drezner, Z. Heuristic Algorithms for the Solution of the Quadratic Assignment Problem, Journal of Applied Mathematics and Decision Sciences, 6, pp. 163-173. 2002.

Drezner, Z. Lower Bounds based on Linear Programming for the Quadratic Assignment Problem, Computational Optimization and Applications, 4 (2), pp. 159-165. 1995.

Drezner, Z. The Extended Concentric Tabu for the Quadratic Assignment Problem, European Journal of Operational Research, 160, pp. 416-422. 2005.

Drezner, Z., Hahn, P.M. and Taillard, E.D. Recent Advances for the QAP Problem with Special Emphasis on Instances that are difficult for Metaheuristic Methods, Annals of Operations Research, 139, pp. 65-94. 2005.

Duong, T.A. and Lam, K.H. Combining Constraint Progamming and Simulated Annealing on University Exam Timetabling. In Proc. 2nd International Conference in Computer Sciences, Research, Innovation and Vision for Future, February 2004, Hanoi, Vietnam, pp. 205-210.

Eiselt, H.A. and Laporte, G. Combinatorial Optimization Problems with Soft and Hard Requirements, The Journal of the Operational Research Society, 38 (9), pp. 785-795. 1987.

Elshafei, A.N. Hospital Layout as a Quadratic Assignment Problem, Operations Research Quarterly, 28 (1), pp. 167-179. 1977.

Ergül, A. GA Based Examination Scheduling Experience at Middle East Technical University. In Practice and Theory of Automated Timetabling: First International Conference, Lecture Notes in Computer Science, Vol. 1153, ed by E.K. Burke and P. Ross, pp. 212-226. New York: Springer. 1996.

Even, S., Itai, A., and Shamir, A. On the Complexity of Timetable and Multicommodity Flow Problems, SIAM Journal of Computing, 5 (4), pp. 691-703. 1976.

Fedjki, C.A. and Duffuaa, S.O. An Extreme Point Algorithm for a Local Minimum Solution to the Quadratic Assignment Problem, European Journal of Operational Research, 156(3), pp. 566-578. 2004.

Feng, T.L., Cheng, Y.K. and Ching, C.H. Applying the Genetic Approach to Simulated Annealing in Solving Some NP-Hard Problems, IEEE Transactions on Systems, Man, and Cybernetics, 23 (6), pp. 1752-1767. 1993.

Feo, T.A. and Bard, J.F. Flight Scheduling and Maintenance Base Planning, Management Science, 35, pp. 1415-1432. 1989.

Feo, T.A. and Resende, M.G.C. Greedy Randomized Adaptive Search Procedures, Journal of Global Optimization, 6, pp. 109-133. 1995.

Filho, G.R. and Lorena, L.A.N. Constructive Genetic Algorithm and Column Generation: an Application to Graph Coloring. In Proc. Of APORS 2000 – The Fifth Conference of the Association of Asian-Pacific Operations Research Societies within IFORS. 2000.

Fisher, M.L. The Lagrangian Relaxation Method for Solving Integer Programming Problems, Management Science, 27 (1), pp. 1-18. 1981.

Fox, B.L. Integrating and Accelerating Tabu Search, Simulated Annealing, and Genetic Algorithms, Annals of Operations Research, 41 (2), pp. 47-67. 1993,

Frieze, A.M. and Yadegar, J. On the Quadratic Assignment Problem, Discrete Applied Mathematics, 5, pp. 89-98. 1983.

Garey, M.R. and Johnson, D.S. The Rectilinear Steiner Tree problem is NP-Complete, SIAM Journal on Applied Mathematics, 32, pp. 826-834. 1977.

Garey, M.R. and Johnson, D.S. Computers and Intractability: A Guide to the Theory of NP-Completeness. San Francisco: W.H. Freeman. 1979.

Gass, S.I. and Harris, C.M. Encyclopedia of Operations Research and Management Science. New York: Springer. 1996.

Gendreau, M., Laporte, G. and Potvin, J.Y. Metaheuristics for the Vehicle Routing Problem. GERAD Research Report G-98-52, Montr Teal, Canada. 1998.

Gendreau, M., Laporte, G. and Potvin, J.Y. Metaheuristics for the Capacitated Vehicle Routing Problem. In The Vehicle Routing Problem, ed by P. Toth and D. Vigo, pp. 129-154. Philadelphia: SIAM. 2001.

Geoffrion, A. Lagrangian Relaxation for Integer Programming, Mathematical Programming Studies, 2, pp. 82-114. 1974.

Geoffrion, A.M. and Graves, G.W. Scheduling Parallel Production Lines with Changeover Costs: Practical Applications of a Quadratic Assignment/LP Approach, Operations Research, 24, pp. 595-610. 1976.

Gilmore, P.C. Optimal and Suboptimal Algorithms for the Quadratic Assignment Problem, SIAM Journal on Applied Mathematics, 10, pp. 305-313. 1962.

Glassey, C.R. and Mizrach, M. A Decision Support System for Assigning Classes to Rooms, Interfaces, 16 (5), pp. 92-100. 1986.

Glover, F. Future Paths for Integer Programming and Links to Artificial Intelligence, Computers and Operations Research, 13, pp. 533-549. 1986.

Glover, F. Tabu Search – Part I, ORSA Journal on Computing, 1, pp. 190-206. 1989.

Glover, F. Tabu Search – Part II, ORSA Journal on Computing, 2, pp. 4-32. 1990.

Gosselin, K. and Truchon, M. Allocation of Classrooms by Linear Programming, The Journal of the Operational Research Society, 37(6), pp. 561-569. 1986.

Graves, R.L., Schrage, L. and Sankaran, J. An Auction Method for Course Registration, Interfaces, 23 (5), pp. 81-92. 1993.

Grötschel, M. Discrete Mathematics in Manufacturing. In Proc. 2nd International Conference on Industrial and Applied Mathematics, July 1991, Washington, USA, pp. 119-145.

Gunawan, A. Applying Metaheuristics to some Timetabling Problems. M.Eng Thesis, National University of Singapore. 2004.

Gunawan, A. Ng, K.M. and Ong, H.L. A Genetic Algorithm for the Teacher assignment Problem for a University in Indonesia, International Journal of Information and Management Sciences, 19 (1), pp. 1-16. 2008.

Gunawan, A. Ong, H.L. and Ng, K.M. Applying Metaheuristics for the Course Scheduling Problem. In Proc. 5th Asia-Pacific Industrial Engineering and Management Systems Conference, December 2004, Gold Coast, Australia.

Gutin, G. and Yeo, A. Polynomial Approximation Algorithms for TSP and QAP with a Factorial Domination Number, Discrete Applied Mathematics, 119 (1-2), pp. 107-116. 2002.

Hansen, P. and Lih, K.-W. Improved Algorithms for Partitioning Problems in Parallel, Pipelined, and Distributed Computing, IEEE Transactions on Computers, 41 (6), pp/ 769-771. 1992.

Harwood, G.B. and Lawless, R.W. Optimizing Faculty Teaching Schedules, Decision Science, 6, pp. 513-524. 1975.

Heffley, D.R. Decomposition of the Koopmans-Beckmann Problem, Regional Science and Urban Economics, 10 (4), pp. 571-580. 1980.

Held, M. and Karp, R.M. The Traveling Salesman Problem and Minimum Spanning Trees, Operations Research, 18, pp. 1138-1162. 1970.

Hertz, A. Finding a Feasible Course Schedule using Tabu Search, Discrete Applied Mathematics, 35(3), pp. 255-270. 1992.

Hertz, A. Tabu Search for Large Scale Timetabling Problems, European Journal of Operational Research, 54(1), pp. 39-47. 1991.

Hillier, F.S. and Connors, M.M. Quadratic Assignment Problem Algorithms and the Location of Indivisible Facilities, Management Science, 13(1), pp. 42-57. 1966.

Johnson, D. A Database Approach to Course Timetabling, The Journal of the Operational Research Society, 44 (5), pp. 425-433. 1993.

Johnson, D. Timetabling University Examinations, Journal of the Operational Research Society, 41, pp. 39-47. 1990.

Joubert, J.W. and Claasen, S.J. A Sequential Insertion Heuristic for the Initial Solution to a Constrained Vehicle Routing Problem, Orion: Operations Research in South Africa, 22(1), pp. 105-116. 2006.

Kapov, J.S. Tabu Search Applied to the Quadratic Assignment Problem, ORSA Journal on Computing, 2(1), pp. 33-45. 1990.

Karisch, S.E. and Rendl, F. Lower Bounds for the Quadratic Assignment Problem via Triangle Decompositions, Mathematical Programming, 71 (2), pp. 137-152. 1995.

Karp, R.M. Reducibility among Combinatorial Problems. In Complexity of Computer Computations, ed by R.E. Miller and J.W. Thatcher, pp. 85-103. New York: Plenum. 1972.

Kim, J.U. and Kim, Y.D. A Lagrangian Relaxation Approach to Multi-Period Inventory/Distribution Planning, Journal of the Operational Research Society, 51, pp. 364-370. 2000.

Kirkpatrick, S., Gellatt, C.D. and Vecchi, M.P. Optimization by Simulated Annealing, Science, 220, pp. 671-680. 1983.

Koopmans, T.C. and Beckmann, M.J. Assignment Problems and the Location of Economic Activities, Econometrica, 25, pp. 53-76. 1957.

Kwan, R.S.K. Bus and Train Driver Scheduling. In Handbook of Scheduling: Algorithms, Models and Performance, ed by J Leung, chapters 51. Boca Raton: CRC Press. 2004.

Laguna, M. and Gonzalez Velarde, J.L. A Search Heuristic for Just in Time Scheduling in Parallel Machines, Journal of Intelligent Manufacturing, 2, pp. 253-260. 1991.

Laplagne, I., Kwan, R.S.K. and Kwan, A.S.K. A Hybridised Integer Programming and Local Search Method for Robust Train Driver Schedules Planning. In Practice and Theory of Automated Timetabling: 5th International Conference, Lecture Notes in Computer Science, Vol. 3616, ed by E.K. Burke and M. Trick, pp. 71-85. New York: Springer. 2005.

Laporte, G. and Desroches, S. The Problem of Assigning Students to Course Sections in a Large Engineering School, Computers and Operations Research, 13 (4), pp. 387 - 394. 1986.
Lawler, E.L. The Quadratic Assignment Problem, Management Science, 9, pp. 586-599. 1963.

Leong, T.Y. and Yeong, W.Y. A Hierarchical Decision Support System for University Examination Scheduling. Working Paper, National University of Singapore. 1990.

Leong, T.Y. and Yeong, W.Y. Examination Scheduling: A Quadratic Assignment Perspective. In Proc. International Conference on Optimization: Techniques and Applications, April 1987, Singapore, pp. 550 – 558.

Lewis, R. A Survey of Metaheuristic-Based Techniques for University Timetabling Problems, OR Spectrum, 30, pp. 167-190. 2008.

Lewis, R. and Paechter, B. Finding Feasible Timetables using Group-Based Operators, IEEE Transactions on Evolutionary Computation, 11 (3), pp. 397-413. 2007.

Li, Y., Pardalos, P.M., and Resende, M.G.C. A Greedy Randomized Adaptive Search procedure for the Quadratic Assignment Problem. In Quadratic Assignment and Related Problems: DIMAS Workshop, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 16, ed by P.M. Pardalos and H. Wolkowicz, pp. 237-261. 1994.

Lim, A., Rodrigues, B. and Zhang, J. Tabu Search embedded Simulated Annealing for the Shortest Route Cut and Fill Problem, Journal of the Operational Research Society, 56, pp. 816-824. 2005.

Lim, M.H., Yuan, Y. and Omatu, S. Efficient Genetic Algorithms using Simple Genes Exchange Local Search Policy for the Quadratic Assignment Problem, Computational Optimization and Applications, 15, pp. 249 – 268. 2000.

Lim, M.H., Yuan, Y. and Omatu, S. Extensive Testing of a Hybrid Genetic Algorithm for Solving Quadratic Assignment Problems, Computational Optimization and Applications, 23, pp. 47-64. 2002.

Liu, S. and Ong, H.L. A Comparative Study of Algorithms for the Flowshop Scheduling Problem, Asia-Pacific Journal of Operational Research, 19, pp. 205-222. 2002.

Loiola, E.M., de Abreu, N.M.M., Boaventura-Netto, P.O., Hahn, P. and Querido, T. A Survey for the Quadratic Assignment Problem, European Journal of Operational Research, 176, pp. 657-690. 2007.

Loo, E.H., Goh, T.N. and Ong, H.L. A Heuristic Approach to Scheduling University Timetables, Computers and Education, 10 (3), pp. 379-388. 1985.

Lotfi, V. and Cerveny, R. A Final Exam Scheduling Package, Journal of the Operational Research Society, 42, pp. 205-216. 1991.

Maniezzo, V. and Colorni, A. Algodesk: An Experimental Comparison of Eight Evolutionary Heuristics Applied to the Quadratic Assignment Problem, European Journal of Operational Research, 81 (1), pp. 188-204. 1995.

Maniezzo, V. and Colorni, A. The Ant System Applied to the Quadratic Assignment Problem, Knowledge and Data Engineering, 11 (5), pp. 769-778. 1999.

McClure, R.H. and Wells, C.E. A Mathematical Programming Model for Faculty Course Assignments, Decision sciences, 15, pp. 409-420. 1984.

McClure, R.H. and Wells, C.E. On the Approximation of Utility Functions for Faculty Teaching Assignments, Socio-Economic Planning Sciences, 19(3), pp. 153-158. 1985.

McCollum, B.. The Implementation of a Central Timetabling System in a Large British Civic University. In Practice and Theory of Automated Timetabling: Second International Conference, Lecture Notes in Computer Science, Vol. 1408, ed by E. Burke and M.W. Carter, pp. 237-253. New York: Springer. 1998.

Merlot, L.T.G., Boland, N., Hughes, B.D. and Stuckey, P.J. A Hybrid Algorithm for the Examination Timetabling Problem. In Practice and Theory of Automated Timetabling: 4th International Conference, Lecture Notes in Computer Science, Vol. 2740, ed by E.K. Burke and P. De Causmaecker, pp. 167-180. New York: Springer. 2003.

Mills, P., Tsang, E. and Ford, J. Applying an Extended Guided Local Search to the Quadratic Assignment Problem, Annals of Operations Research, 118 (1-4), pp. 121-135. 2003.

Miranda, G., Luna, H.P.L., Mateus, G.R. and Ferreira, R.P.M. A Performance Guarantee Heuristic for Electronic Components Placement Problems including Thermal Effects, Computers and Operations Research, 32, pp. 2937-2957. 2005.

Moody, D., Kendall, G. and Bar-Noy, A. Constructing Initial Neighborhoods to Identify Critical Constraints. In Proc. 7th International Conference on the Practice and Theory of Automated Timetabling, August 2008, Montreal, Canada.

Naji Azimi, Z. Hybrid Heuristics for Examination Timetabling Problem, Applied Mathematics and Computation, 163, pp. 705-733. 2005.

Nowicki, E. and Smutnicki, C. A Fast Taboo Search Algorithm for the Job Shop Problem, Management Science, 42(2), pp. 797-813. 1996.

Osman, I.H. and Laporte, G. Metaheuristics: a Bibliography, Annals of Operations Research, 63, pp. 513-623. 1996.

Paquete, L. and Stutzle, T. Empirical Analysis of Tabu Search for the Lexicographic Optimization of the Examination Timetabling Problem. In Proc. 4th International Conference on Practice and Theory of Automated Timetabling, August 2002, Gent, Belgium, pp. 413-420.

Peng, T., Huanchen, W. and Dongme, Z. Simulated Annealing for the Quadratic Assignment Problem: A Further Study, Computers and Industrial Engineering, 31 (3-4), pp. 925-928. 1996.

Petrovic, S. and Burke, E.K. University Timetabling. In the Handbook of Scheduling: Algorithms, Models and Performance Analysis, ed. By J. Leung. CRC Press. 2004.

Petrovic, S., Patel, V. and Yong, Y. Examination Timetabling with Fuzzy Constraints. In Practice and Theory of Automated Timetabling: 5th International Conference, Lecture Notes in Computer Science, Vol. 3616, ed by E.K. Burke and M. Trick, pp. 313-333. New York: Springer. 2005.

Plateau, A., Tachat, D. and Tolla, P. A Hybrid Search Combining Interior Point Methods and Metaheuristics for 0-1 Programming, International Transactions in Operational Research, 9, pp. 731–746. 2002.

Prais, M. and Ribeiro, C.C. Reactive GRASP: An Application to a Matrix Decomposition Problem in TDMA Traffic Assignment, INFORMS Journal on Computing, 12, pp. 164-176. 2000.

Puchinger, J. and Raidl, G.R. Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: a Survey and Classification. In Artificial Intelligence and Knowledge Engineering Applications: First International Work-Conference on the Interplay between Natural and Artificial Computation, Lecture Notes in Computer Science, Vol. 3562, ed by J. Mira and J.R. Alvarez, pp. 41-53. Berlin: Springer. 2005.

Puchinger, J., Raidl, G.R. and Koller, G. Solving a Real-World Glass Cutting Problem. In EvoCOP 2004, Lecture Notes in Computer Science, Vol. 3004, ed by J. Gottlieb and G.R. Raidl, pp. 165-176, New York: Springer, 2004.

Qu, R., Burke, E.K., McCollum, B., Merlot, L.T.G. and Lee, S.Y. A Survey of Search Methodologies and Automated Approaches for Examination Timetabling. Computer Science Technical Report No. NOTTCS-TR-2006-4, University of Nottingham. 2006.

Radhakrishnan, S. and Ventura, J.A. Simulated Annealing for Parallel Machine Scheduling with Earliness-Tardiness Penalties and Sequence-Dependent Set-Up Times, International Journal of Production Research, 38 (10), pp. 2233-2252. 2000.

Rahoual, M. and Saad, R. Solving Timetabling Problems by Hybridizing Genetic Algorithms and Tabu Search. In Proc. 6th International Conference on Practice and Theory of Automated Timetabling, August 2006, Brno, Czech Republic, pp. 467-472.

Rendl, F. and Sotirov, R. Bounds for the Quadratic assignment Problems using the Bundle Method, Mathematical Programming, 109 (2-3), pp. 505-524. 2007.

Resende, M.G.C., Ramakrishnan, K.G. and Drezner, Z. Computing Lower Bounds for the Quadratic Assignment Problem with an Interior Point Algorithm for Linear Programming, Operations Research, 43, pp. 781-791. 1995.

Ross, A. A Two-Phased Approach to the Supply Network Reconfiguration Problem, European Journal of Operational Research, 122, pp. 18-30. 2000.

Rossin, D.F., Springer, M.C. and Klein, B.D. New Complexity Measures for the Facility Layout Problem: An Empirical Study using Traditional and Neural Network Analysis, Computers and Industrial Engineering, 36 (3), pp.585-602. 1999.

Sabin, G.C.W and Winter, G.K. The Impact of Automated Timetabling on Universities – A Case Study, Journal of the Operational Research Society, 37 (7), pp. 689-693. 1986.

Sahni, S. and Gonzales, T. P-Complete Approximation Problems, Journal of the Association for Computing Machinery, 23, pp. 555-565. 1976.

Saleh Elmohamed, M.A., Coddington, P. and Fox, G. A Comparison of Annealing Techniques for Academic Course Scheduling. In Practice and Theory of Automated Timetabling: Second International Conference, Lecture Notes in Computer Science, Vol. 1408, ed by E. Burke and M.W. Carter, pp. 92-112. New York: Springer. 1998.

Sampson, S.E., Freeland, J.R. and Weiss, E.N. Class Scheduling to Maximize Participant Satisfaction, Interfaces, 25 (1), pp. 30-41. 1995.

Schaerf, A. and Meisels, A. Solving Employee Timetabling Problems by Generalized Local Search. In AI*IA 99: Advances in Artificial Intelligence: 6th Congress of the Italian Association for Artificial Intelligence, Lecture Notes in Computer Science, Vol. 1792, ed. E. Lamma and P. Mello, pp. 380-389. Berlin: Springer. 2000.

Schaerf. A Survey of Automated Timetabling, Artificial Intelligence Review, 13, pp. 87-127. 1999.

Schniederjans, M.J. and Kim, G.C. A Goal Programming Model to Optimize Departmental Preference in Course Assignments, Computers and Operations Research, 14 (2), pp. 87-96. 1987.

Schönberger, J., Mattfeld, D.C. and Kopfer, H. Memetic Algorithm Timetabling for Non-commercial Sport Leagues, European Journal of Operational Research, 153 (1), pp. 102-116. 2004.

Selim, S.M. An Algorithm for Constructing a University Faculty Timetable, Computers and Education, 6 (4), pp. 323 – 334. 1982.

Sheibani, K. An Evolutionary Approach for the Examination Timetabling Problems. In Proc. 4th International Conference on Practice and Theory of Automated Timetabling, August 2002, Gent, Belgium, pp. 387-396.

Siu, F. and Chang, R.K.C. Effectivenes of Optimal Node Assignments in Wavelength Division Multiplexing Networks with Fixed Regular Virtual Topologies, Computer Networks, 38 (1), pp. 61-74. 2002.

Socha, K., Knowles, J. and Samples, M. A Max-Min Ant System for the University Course Timetabling Problem. In ANTS 2002, Lecture Notes in Computer Science, Vol. 2463, ed by M. Dorigo, pp. 1-13, New York: Springer. 2002.

Socha, K., Samples, M. and Manfrin, M. Ant Algorithms for the University Course Timetabling Problem with regards to the State-of-the-Art, IRIDIA, Universite Libre de Bruxelles, CP 194/6, Av. Franklin D. Roosevelt 50, 1050 Bruxelles, Belgium (http://iridia.ulb.ac.be email: {ksocha|msampels|mmanfrin}@ulb.ac.be). 2003.

Stallaert, J. Automated Timetabling Improves Course Scheduling at UCLA, Interfaces, 27 (4), pp. 67-81. 1997.

Steinberg, L. The Background Wiring Problem: A Placement Algorithm, SIAM Review, 3, pp. 37-50. 1961.

Taillard, É.D. Robust Taboo Search for the Quadratic Assignment Problem, Parallel Computing, 17, pp. 443-455. 1991.

Taillard, É.D. and Gambardella, L.M. Adaptive Memories for the Quadratic Assignment Problem. Technical Report IDSIA-87-97, IDSIA, Lugano, Switzerland. 1997.

Taillard, É.D. Robust Taboo Search for the Quadratic Assignment Problem, Parallel Computing, 17, pp. 443-455. 1991.

Talbi, E.G. A Taxonomy of Hybrid Metaheuristics, Journal of Heuristics, 8, pp. 541-564. 2002.

Talbi, E.G., Hafidi, Z. and Geib, J.M. A Parallel Adaptive Tabu Search Approach, Parallel Computing, 24 (14), pp. 2003-2019. 1998.

ten Eikelder, H.M.M. and Willemen, R.J. Some Complexity Aspects of Secondary School Timetabling Problems. In Practice and Theory of Automated Timetabling: 3rd International Conference, Lecture Notes in Computer Science, Vol. 2079, ed by E. Burke and W. Erben, pp. 18-27. New York: Springer. 2001.

Thompson, J. and Dowsland, K. A Robust Simulated Annealing Based Examination Timetabling System, Computers and Operations Research, 25, pp. 637-648. 1998.

Thompson, J. and Dowsland, K.A. General Cooling Schedules for a Simulated Annealing Based Timetabling System. In Practice and Theory of Automated Timetabling: First International Conference, Lecture Notes in Computer Science, Vol. 1153, ed by E.K. Burke and P. Ross, pp. 345-363. New York: Springer. 1996.

Tian, P., Wang, H.C. and Zhang, D.M. Simulated Annealing for the Quadratic Assignment Problem: A Further Study, Computers and Industrial Engineering, 31 (3-4), pp. 925-928. 1996.

Tillett, P.I. An Operations Research Approach to the Assignment of Teachers to Courses, Socio-Economic Planning Sciences, 9, pp. 101-104. 1975.

Tripathy, A. A Lagrangian Relaxation Approach to Course Timetabling, The Journal of the Operational Research Society, 31 (7), pp. 599 – 603. 1980.

Tripathy, A. Computerised Decision Aid for Timetabling – A Case Analysis, Discrete Applied Mathematics, 35, pp. 313–323. 1992.

Tseng, L.Y. and Liang, S.C. A Hybrid Metaheuristic for the Quadratic Assignment Problem, Computational Optimization and Applications, 34, pp. 85-113. 2006.

Tseng, L.Y. and Lin, Y.T. A Hybrid Genetic Local Search Algorithm for the Permutation Flowshop, European Journal of Operational Research, 198, pp. 84-92. 2009.

Ueda, H., D. Ouchi, K. Takahashi and T. Miyahara. A Co-evolving Timeslot/Room Assignment Genetic Algorithm Technique for University Timetabling. In Practice and Theory of Automated Timetabling: 3rd International Conference, Lecture Notes in Computer Science, Vol. 2079, ed by E. Burke and W. Erben, pp. 48-63. New York: Springer. 2001.

Valdes, R.A., Crespo, E. and Tamarit, J.M. Assigning Students to Course Sections using Tabu Search, Annals of Operations Research, 96, pp. 1-16. 2000.

Valdes, R.A., Crespo, E. and Tamarit, J.M. Design and Implementation of a Course Scheduling System using Tabu Search, European Journal of Operational Research, 137, pp. 512–523. 2002.

Valdes, R.A., Martin, G. and Tamarit, J.M. Constructing Good Solutions for the Spanish School Timetabling Problem, The Journal of the Operational Research Society, 47 (10), pp. 1203–1215. 1996.

Van den Broek, J., Hurkens, C. and Woeginger, G. Timetabling Problems at the TU Eindhoven. In Practice and Theory of Automated Timetabling: 6th International Conference, Lecture Notes in Computer Science, Vol. 3867, ed by E.K. Burke and H. Rudova, pp. 210-227. Berlin: Springer. 2007.

Van Laarhoven, P.J.M., Aarts, E.H.L and Lenstra, J.K. Job Shop Scheduling by Simulated Annealing, Operations Research, 40, pp. 113-125. 1992.

Voss, S., Martello, S., Osman, I.H. and Roucairol, C. Meta-heuristics-Advances and Trends in Local Search Paradigms for Optimisation. Boston, Mass: Kluwer Academics Publisher. 1999.

Wang, Y.Z. An Application of Genetic Algorithm Methods for Teacher Assignment Problems, Expert Systems with Applications, 22 (4), pp. 295-302. 2002.

Weare, R., Burke, E.K. and Elliman, D. A Hybrid Genetic Algorithm for Highly Constrained Timetabling Problems. Computer Science Technical Report No. NOTTCS-TR-1995–8, University of Nottingham. 1995.

West, D.H. Algorithm 608: Approximate Solution of the Quadratic Assignment Problem, ACM Transactions on Mathematical Software, 9, pp. 461-466. 1983.

White, G.M. and Chan, P.W. Towards the Construction of Optimal Examination Schedules, INFOR, 17 (3), pp. 219-229. 1979.

White, G.M. and Wong, S.K.S. Interactive Timetabling in Universities, Computers and Education, 12 (4), pp. 521–529. 1988.

White, G.M. and Xie, B.S. Examination Timetables and Tabu Search with Longer-Term Memory. In Practice and Theory of Automated Timetabling: 3rd International Conference, Lecture Notes in Computer Science, Vol. 2079, ed by E. Burke and W. Erben, pp. 85-103. New York: Springer. 2001.

White, G.M. and Zhang, J. Generating Complete University Timetables by Combining Tabu Search with Constraint Logic. In Practice and Theory of Automated Timetabling: Second International Conference, Lecture Notes in Computer Science, Vol. 1408, ed by E.K. Burke and M.W. Carter, pp. 187-198. New York: Springer. 1998.

Wilhelm, M.R. and Ward, T.L. Solving Quadratic Assignment Problem by Simulated Annealing, IIE Transactions, 19 (1), pp. 107-119. 1987.

Wren, A. Scheduling, Timetabling and Rostering – A Special Relationship? In Practice and Theory of Automated Timetabling: First International Conference, Lecture Notes in Computer Science, Vol. 1153, ed by E. Burke and P. Ross, pp. 46-75. New York: Springer. 1996.

Wright, M. School Timetabling using Heuristic Search, The Journal of the Operational Research Society, 47 (3), pp. 347-357. 1996.

Yong, L., Pardalos, P.M. and Resende, M.G.C. A Greedy Randomized Adaptive Search procedure for the Quadratic Assignment Problem. In Quadratic Assignment and Related Problems: DIMAS Workshop, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 16, ed by P.M. Pardalos and H. Wolkowicz, pp. 237-261. 1994.

Yu, E and Sung, K.S. A Genetic Algorithm for a University Weekly Courses Timetabling Problem, International Transactions in Operational Research, 9, pp. 703–717. 2002.

Yu, J. and Sarker, B.R. Directional Decomposition Heuristic for a Linear Machine-Cell Location problem, European Journal of Operational Research, 149 (1), pp. 142-184. 2003.

PUBLICATIONS

Gunawan, A., Ng, K.M. and Poh, K.L. A Mathematical Programming Model for a Timetabling Problem. In Proc. International Conference on Scientific Computing, June 2006, Nevada, USA.

Gunawan, A., Ng, K.M. and Poh, K.L. An Improvement Heuristic for the Timetabling Problem. In Proc. 7th Asia-Pacific Industrial Engineering and Management Systems Conference, December 2006, Bangkok, Thailand.

Gunawan, A., Ng, K.M. and Poh, K.L. An Improvement Heuristic for the Timetabling Problem, International Journal Computational Science, 1 (2), pp. 162-178. 2007.

Gunawan, A., Poh, K.L. and Ng, K.M. Design of Educational Timetabling Systems. In Proc. Asia-Pacific Systems Engineering Conference, March 2007, Singapore.

Gunawan, A., Ng, K.M. and Poh, K.L. Solving the Teacher Assignment - Course Scheduling Problem by a Hybrid Algorithm, International Journal of Computer, Information and Systems Science and Engineering, 1 (2), pp. 136-141. 2007.

Gunawan, A., Ng, K.M. and Poh, K.L. Solving the Teacher Assignment - Course Scheduling Problem by a Hybrid Algorithm. In Proc. 12th International Conference on Computer, Information and Systems Science and Engineering, July 2007, Prague, Czech Republic.

Ng, K.M., Gunawan, A. and Poh, K.L. A Hybrid Algorithm for the Quadratic Assignment Problem. In Proc. International Conference on Conference on Scientific Computing, July 2008, Nevada, USA.

Gunawan, A., Ng, K.M. and Poh, K.L. A Hybrid Algorithm for the University Course Timetabling Problem. In Proc. 7th International Conference on the Practice and Theory of Automated Timetabling, August 2008, Montreal, Canada.

Gunawan, A., Ng, K.M. and Poh, K.L. A Hybridized Lagrangian Relaxation and Simulated Annealing for the Course Timetabling Problem. Submitted to European Journal of Operational Research.

Ng, K.M., Gunawan, A. and Poh, K.L. A GRASP-SA-TS Hybrid Algorithm for the Quadratic Assignment Problem. Submitted to Computers and Operations Research.

APPENDIX A Procedures of the Improvement Heuristic

The details of three phases of the improvement heuristic presented in Chapter 4 are described as follows:

Pre-Processing Phase

(1) For each course *j*, construct a list I_j that consists of teachers who are willing to teach course *j* that are sorted in non-increasing order of course preference.

(2) For each teacher *i*, construct a list LM_i of a 2-tuple (day *l* and time period *m*) that consists of days and time periods sorted in non-increasing order of time period preference.

Construction Phase

This phase consists of two sub-phases:

Teacher assignment allocation sub-phase

(3) For each course *j*, find the minimum number of teachers needed, LT_j. The objective in this phase is to find teacher(s) from list I_j with the highest preference without violating the maximum number of courses taught constraint. *Checking procedure* must be applied in this step. If the minimum number of teacher, LT_j is one, allocate the selected teacher to all sections of course *j*, otherwise, distribute the sections to the selected teachers, for instance, if LT_j is two and course *j* has four sections, then each teacher will teach two course sections.

(4) Classify each teacher based on their total number of courses taught. They are divided into three groups: teachers who have number of courses taught more than the maximum value allowed (Group 1), teachers who have number of courses taught less than or equal to the maximum value allowed and greater than one course (Group 2) and teachers who are not allocated to any courses (Group 3).

(5) If $|Group_1|$ and $|Group_3| = 0$, go to Step (7), otherwise go to Step (6).

(6) **Repeat** the following steps **until** the total number of iterations reaches the preset maximum number of iterations or $|Group_1|$ and $|Group_3| = 0$:

(6.1) **If** $|Group_1| \neq 0$, **do**

- Choose one teacher randomly from Group 1 and find another teacher who can teach one of the course

- Classify all teachers again into Groups 1, 2 and 3.

(6.2) **If** $|Group_3| \neq 0$, **do**

- Choose one teacher randomly from Group 3 and allocate a course from his course preference list
- Classify all teachers again into Groups 1, 2 and 3.

Course scheduling sub-phase

(7) Allocate each course section to time periods based on time preference. *Checking procedure* described earlier must be applied.

(8) Check whether any courses are taught more than once a day. If any, keep the course in Excess List 1 (EL_1).

(9) Check whether any time periods have course sections allocated more than the capacity allowed. If any, keep the particular time period in Excess List 2 (EL_2).

(10) Check whether any teachers have their course sections not evenly spread out throughout the week. If any, keep the teacher in the Excess List 3 (EL_3).

(11) If $|EL_1|$, $|EL_2|$ and $|EL_3| = 0$, go to Step (13), otherwise go to Step (12).

(12) **Repeat** the following steps **until** the total number of iterations reaches the preset maximum number of iterations or $|EL_1|$, $|EL_2|$ and $|EL_3| = 0$:

(12.1) **If** $|EL_1| \neq 0$, **do**

- Choose a course randomly from Excess List 1 and find a new day and time periods.
- Update Excess Lists 1, 2 and 3.

(12.2) **If** $|EL_2| \neq 0$, **do**

- Choose a time period randomly from Excess List 2.
- Determine number of course sections have to be reallocated.
- For each selected course section, find a new day and new time periods.
- Update Excess Lists 1, 2 and 3.

(12.3) **If** $|EL_3| \neq 0$, **do**

- Choose a teacher from Excess List 3.
- Find the day taught by that teacher which has number of course sections taught more than the upper bound of number of course sections taught per day.
- Determine number of course sections have to be reallocated.
- For each selected course section, find a new day and new time periods.
- Update Excess Lists 1, 2 and 3.

(13) Calculate the total objective function value

Improvement Phase

(14) Keep the current initial solution as the best solution obtained and treated as the initial solution.

(15) Start from the initial solution.

(16) Choose a course randomly including the teacher allocated to that particular course.

(17) Find another teacher who has the minimum number of courses taught and less than the maximum number of courses taught. If found, go to Step (16), otherwise go to Step (15).

(18) Reallocate the new teacher to the course.

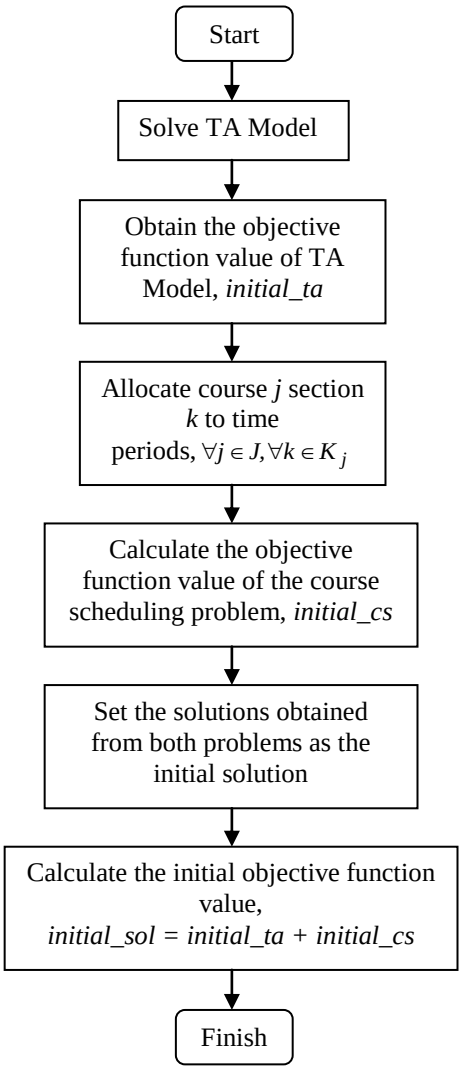
(19) Reallocate the time periods of the course.

(20) Update the total objective function value.

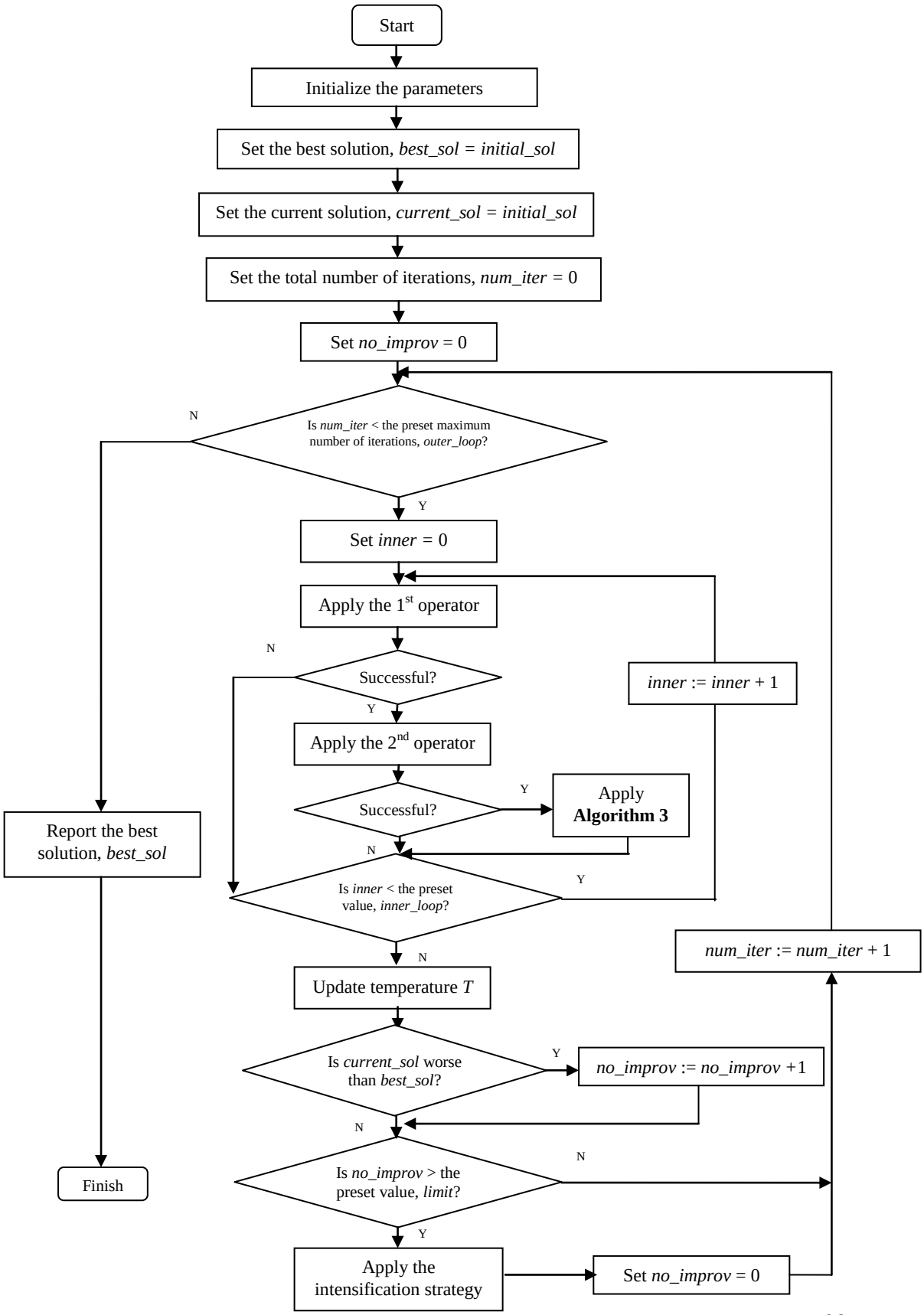
(21) Check whether the final objective function value is better than the best solution obtained. If Yes, keep the current solution as the best solution and update the starting initial solution. Otherwise, go back to the starting initial solution.

(22) Repeat Steps (15) to (21) until the total number of iterations reaches the preset maximum number of iterations.

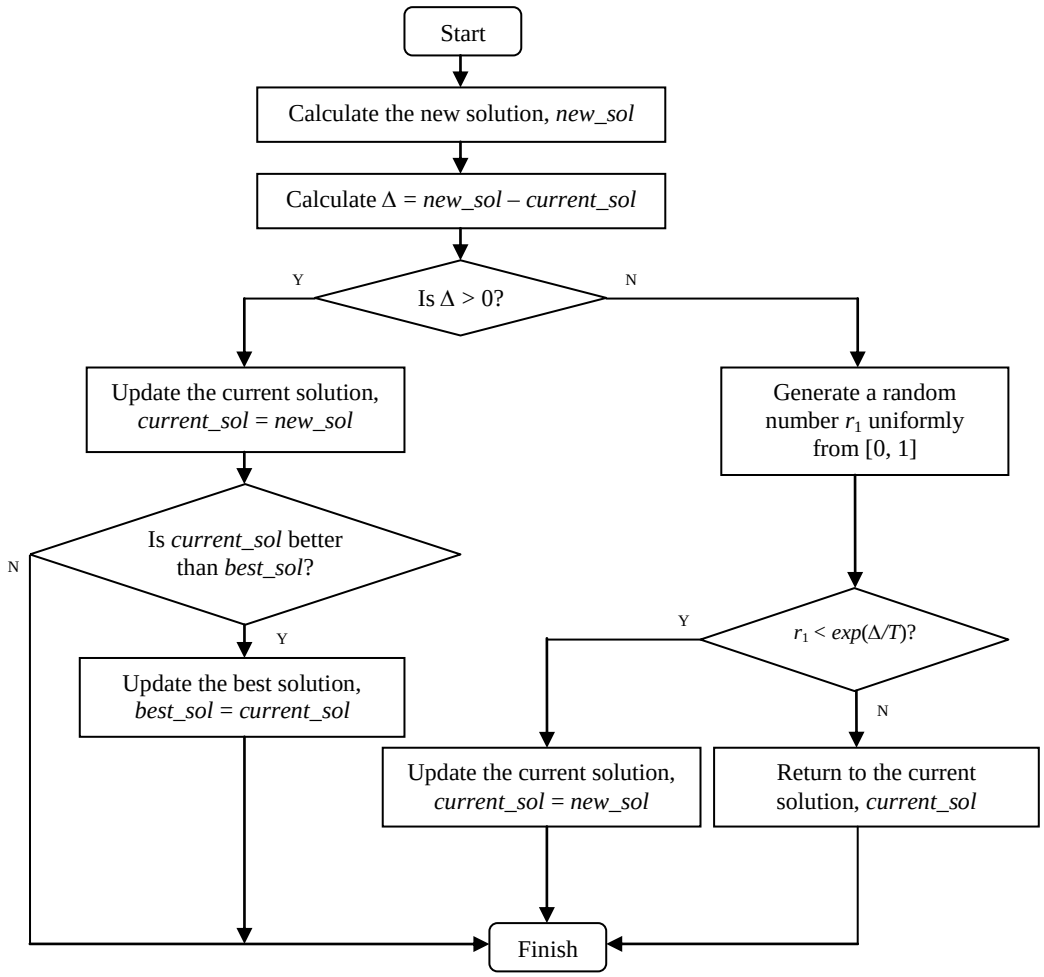
APPENDIX B1 Flow Chart of the Construction Phase



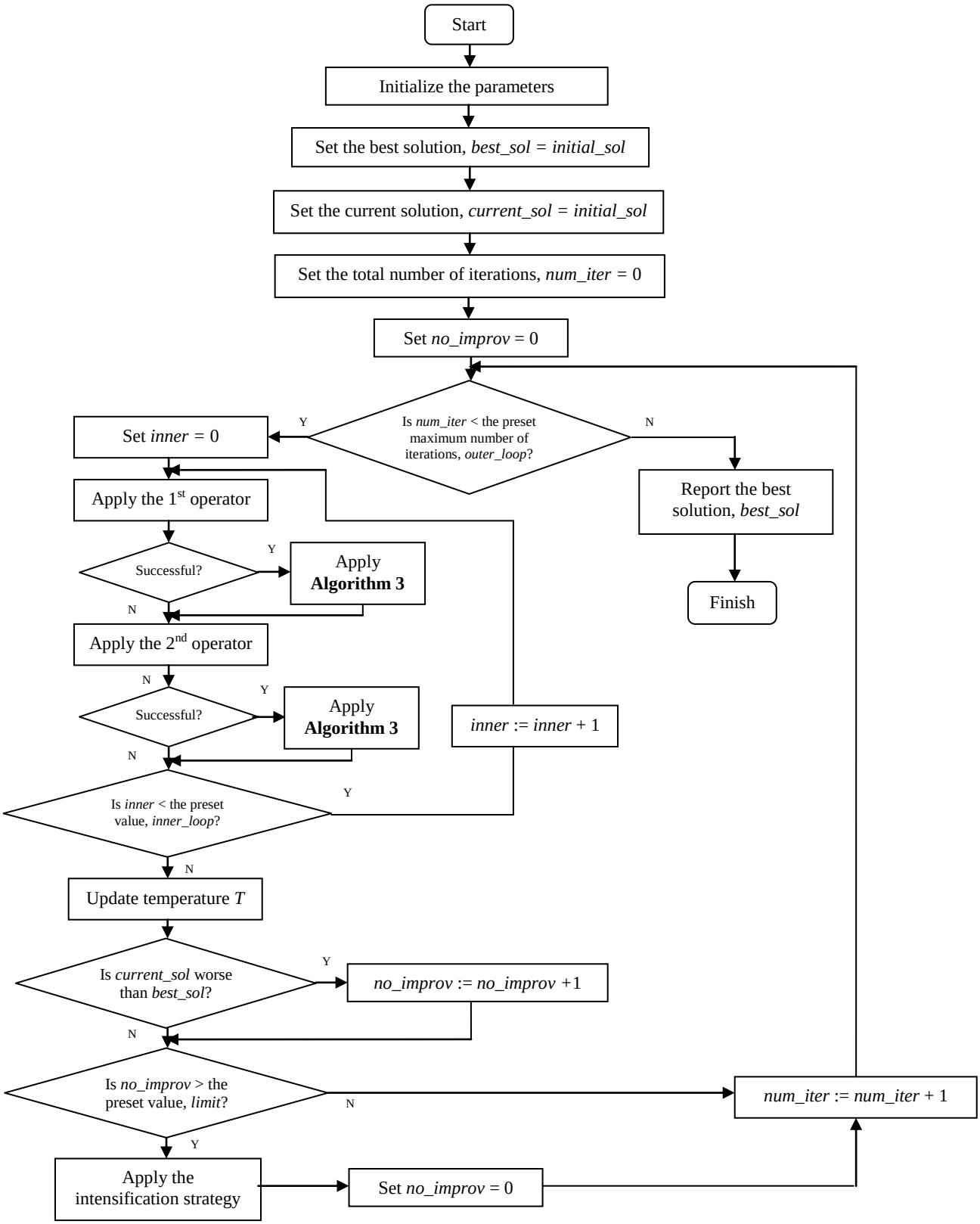
APPENDIX B2 Flow Chart of Algorithm SA1



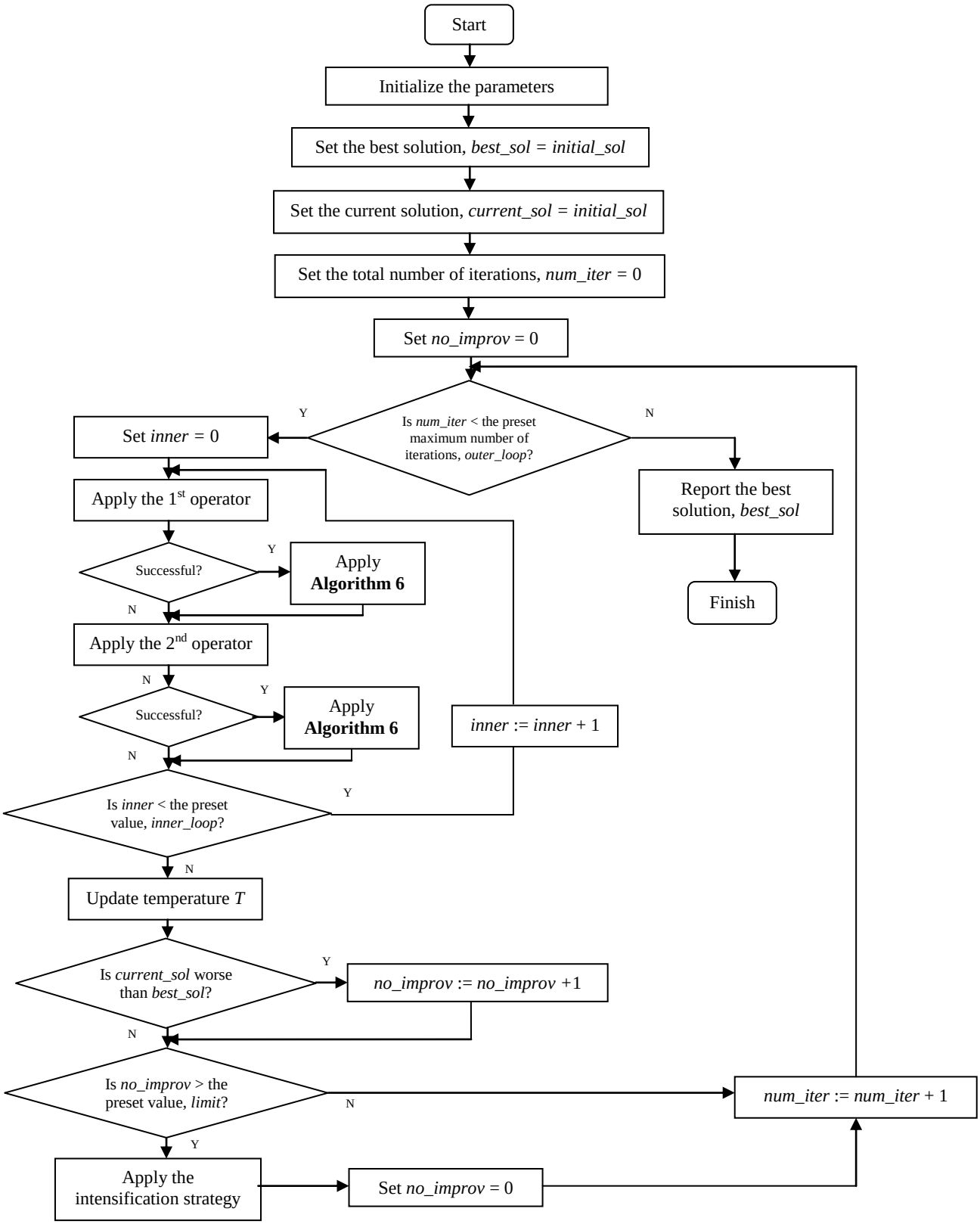
APPENDIX B3 Flow Chart of the Evaluation Process of Algorithms SA1 and SA2 (Algorithm 3)



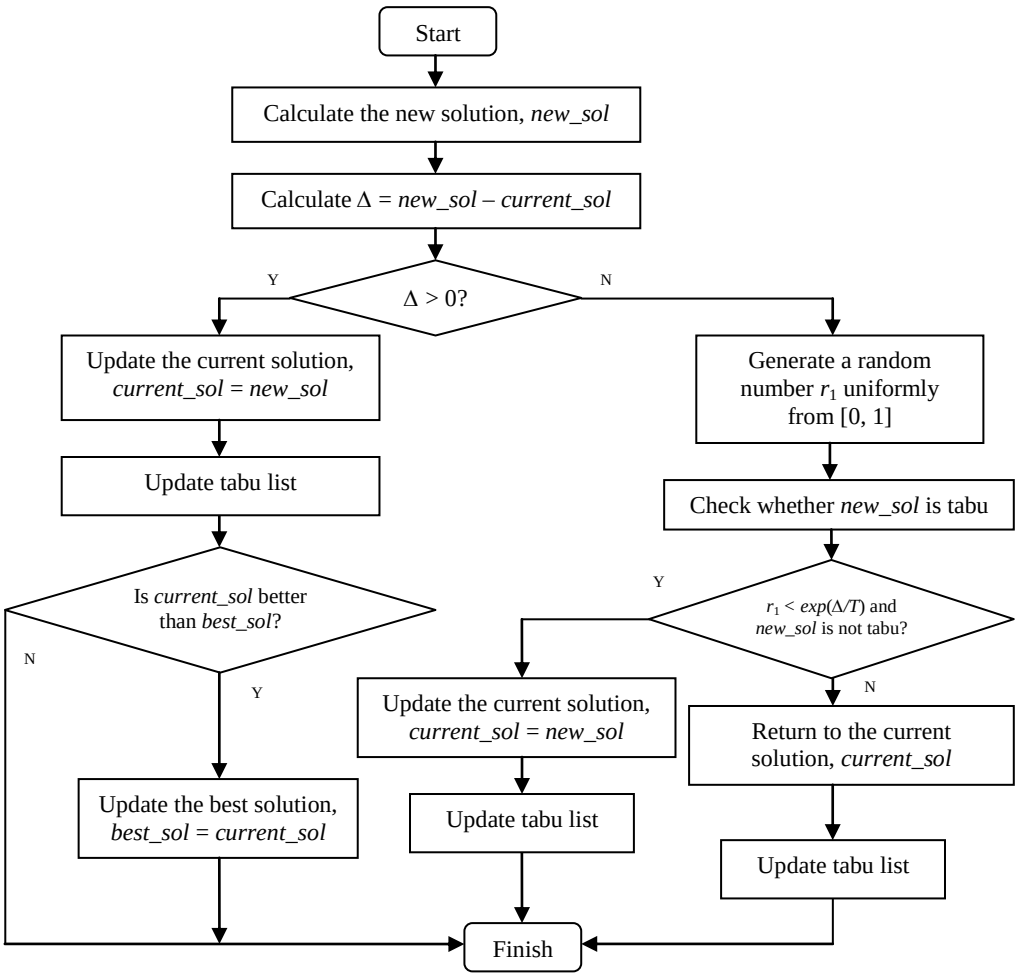
APPENDIX B4 Flow Chart of Algorithm SA2



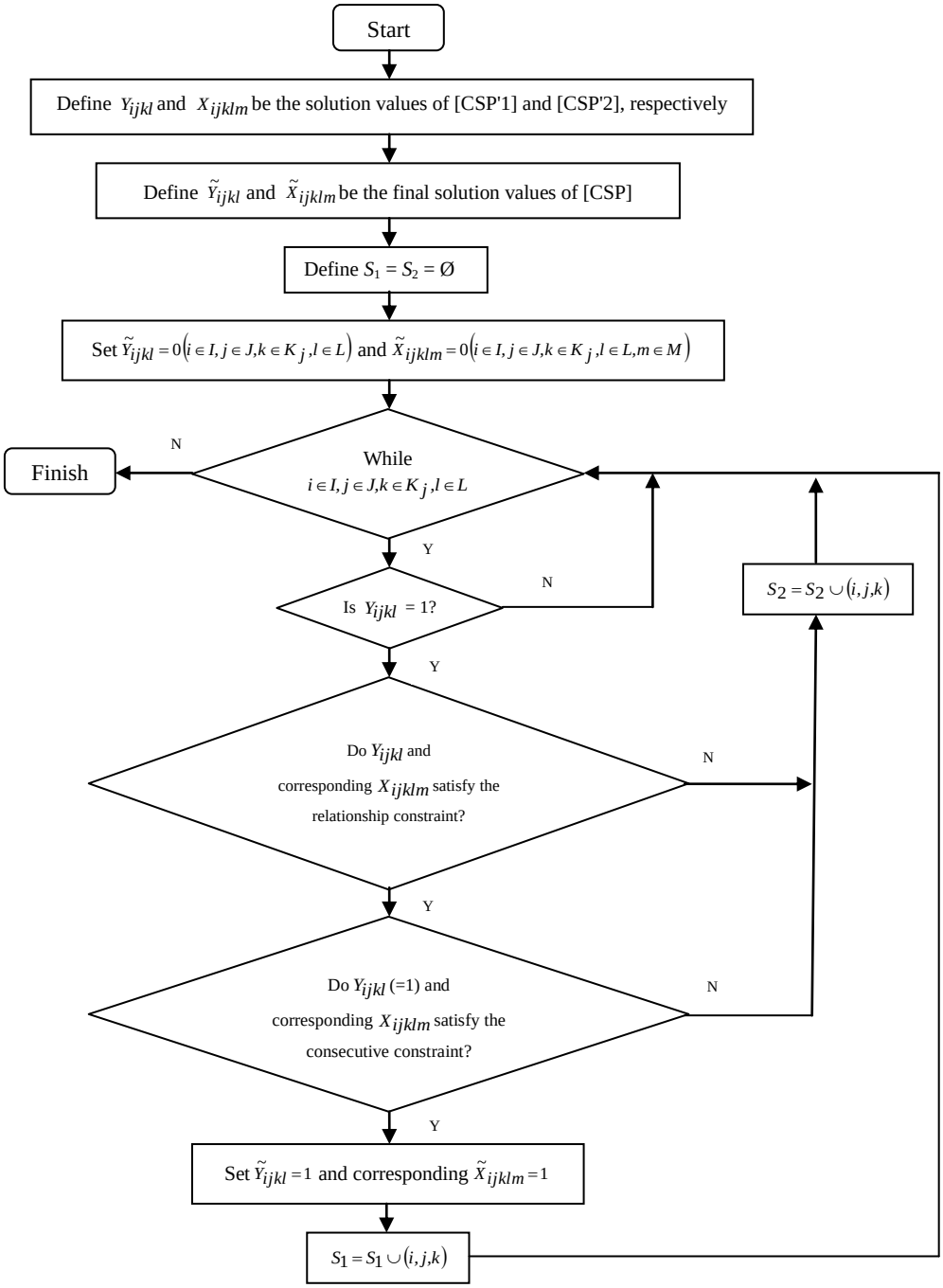
APPENDIX B5 Flow Chart of Algorithm SA-TS



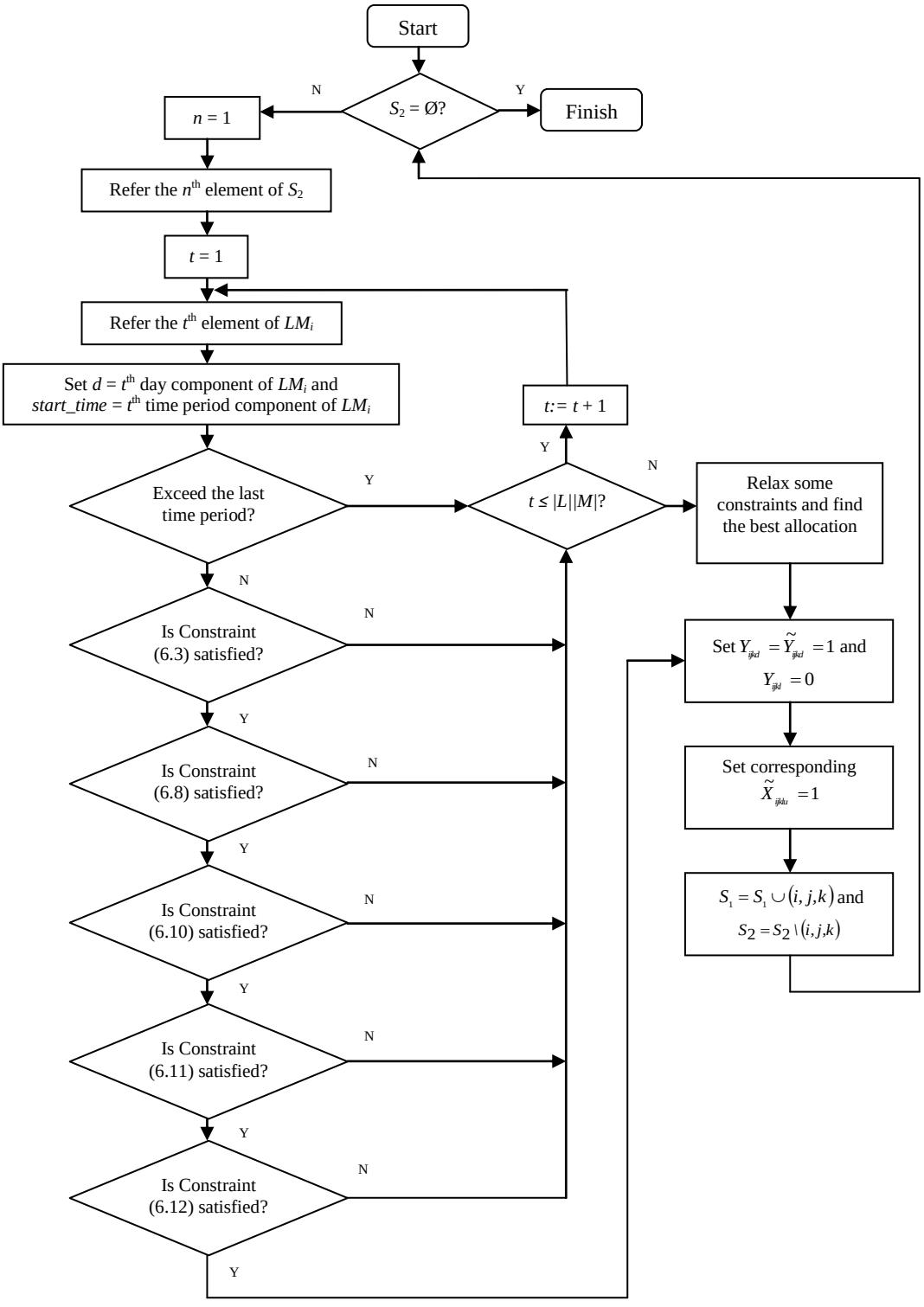
APPENDIX B6 Flow Chart of the Evaluation Process of Algorithm SA-TS (Algorithm 6)



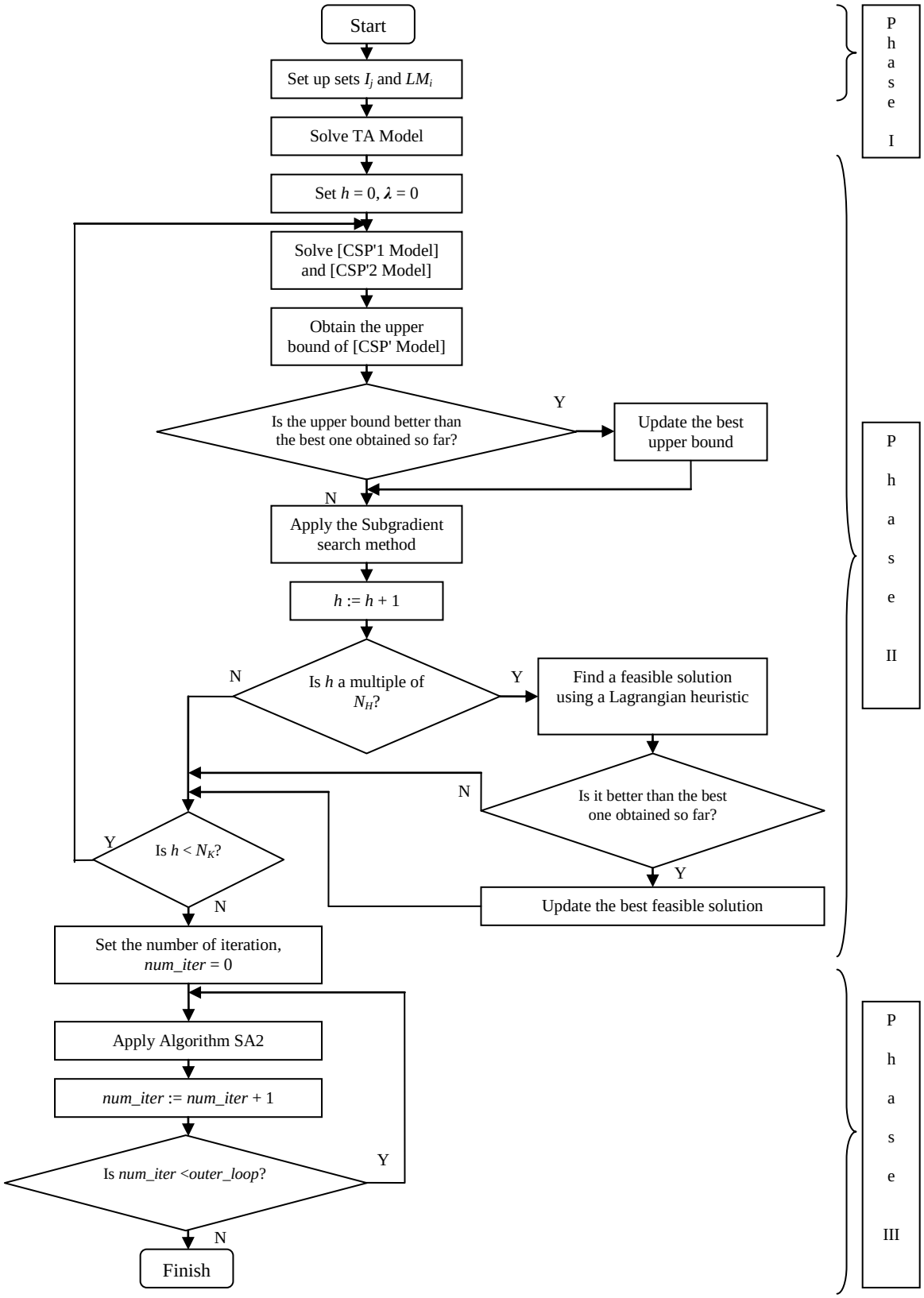
APPENDIX C1 Flow Chart of the First Process of a Lagrangian Heuristic



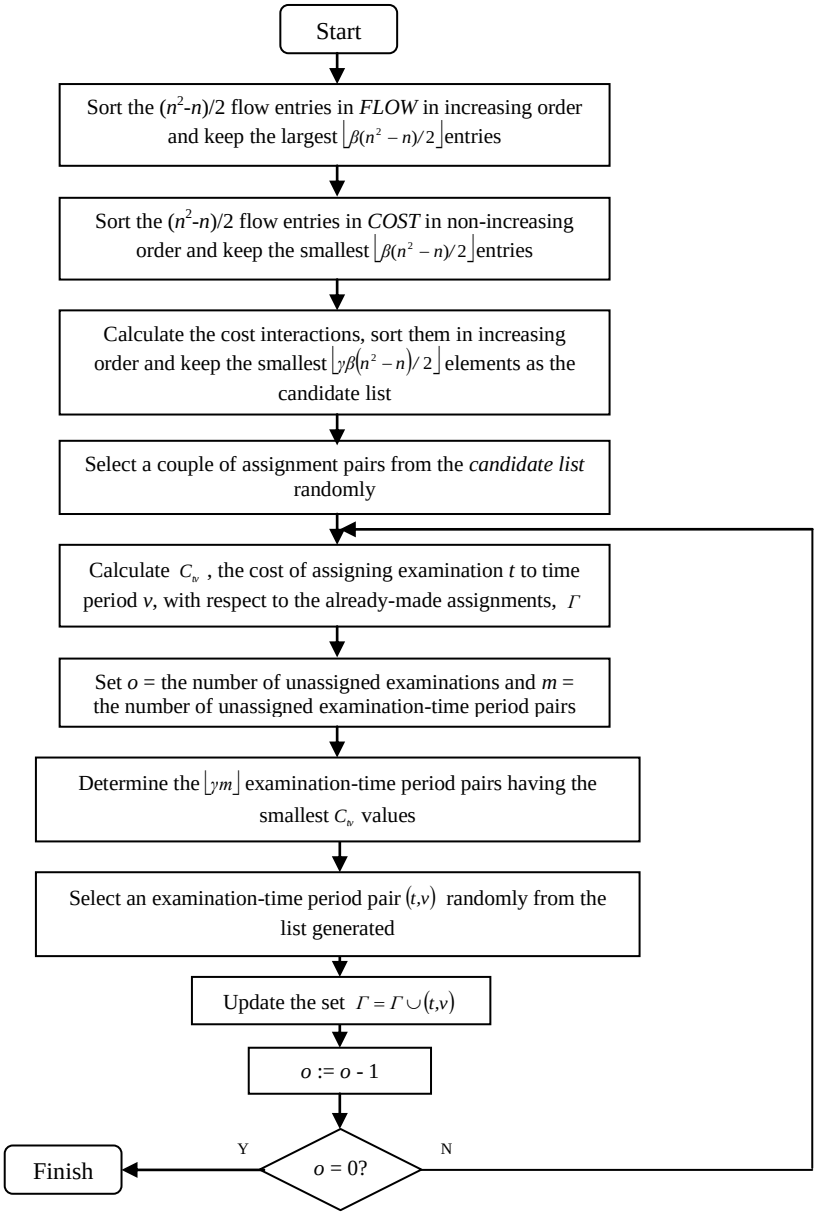
APPENDIX C2 Flow Chart of the Second Process of a Lagrangian Heuristic



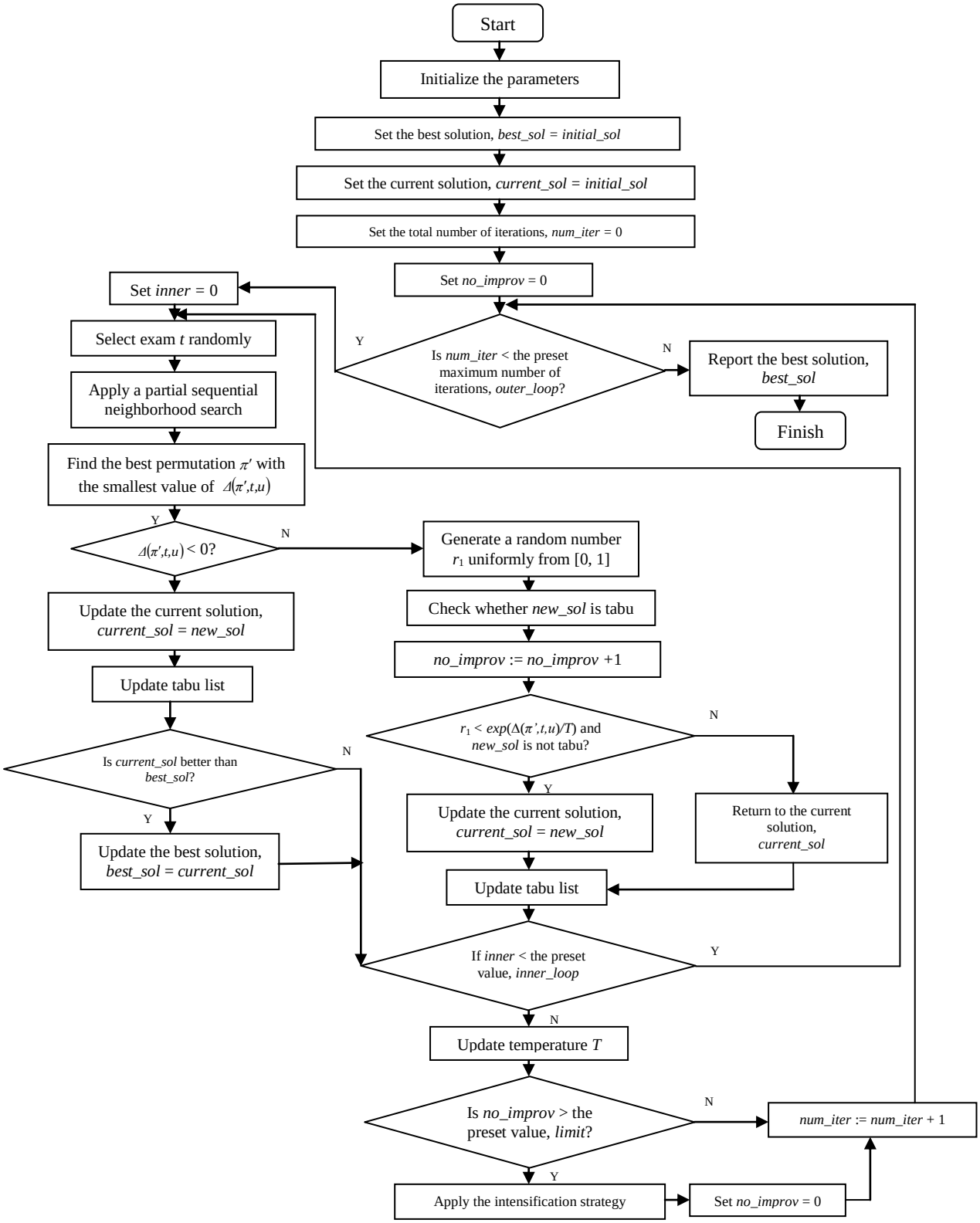
APPENDIX C3 Flow Chart of Algorithm LR-SA



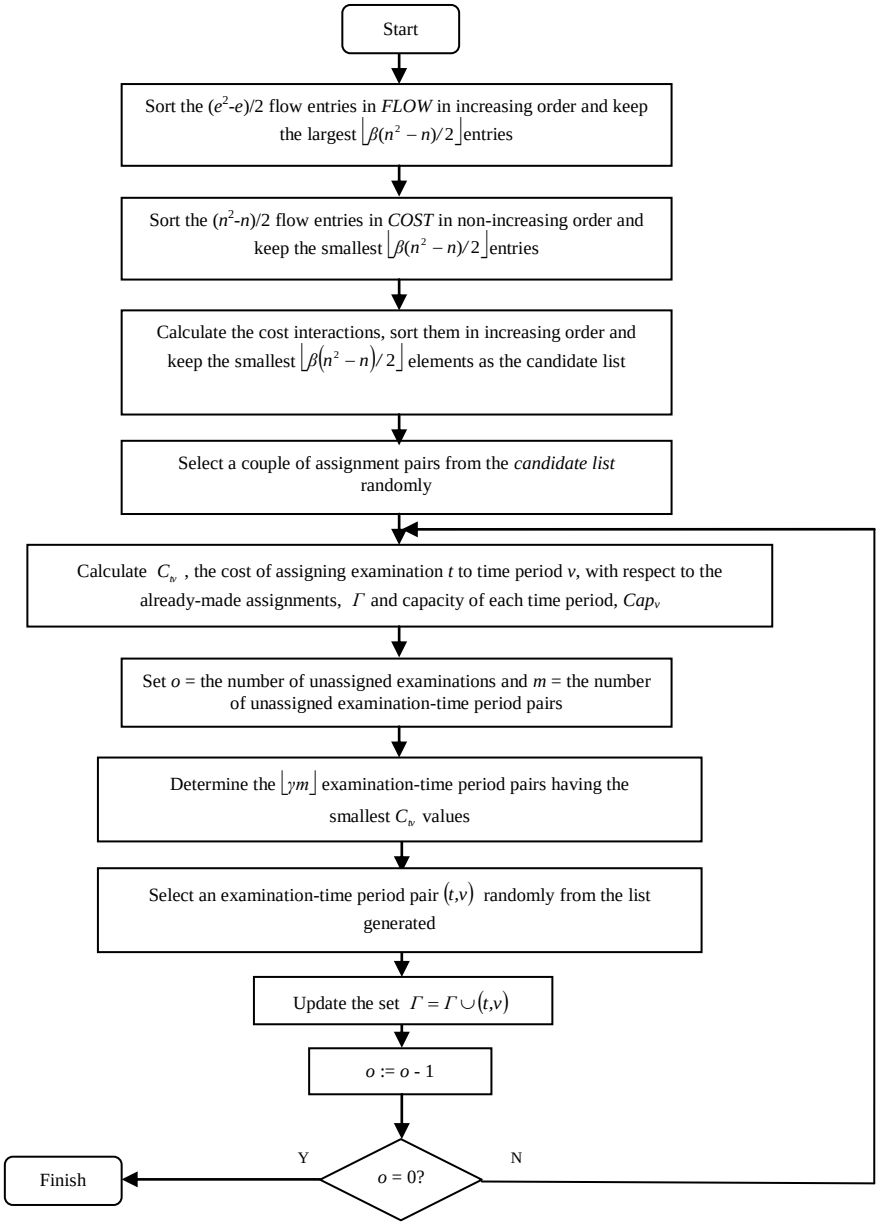
APPENDIX D1 Flow Chart of the Construction Phase of GRASP Algorithm



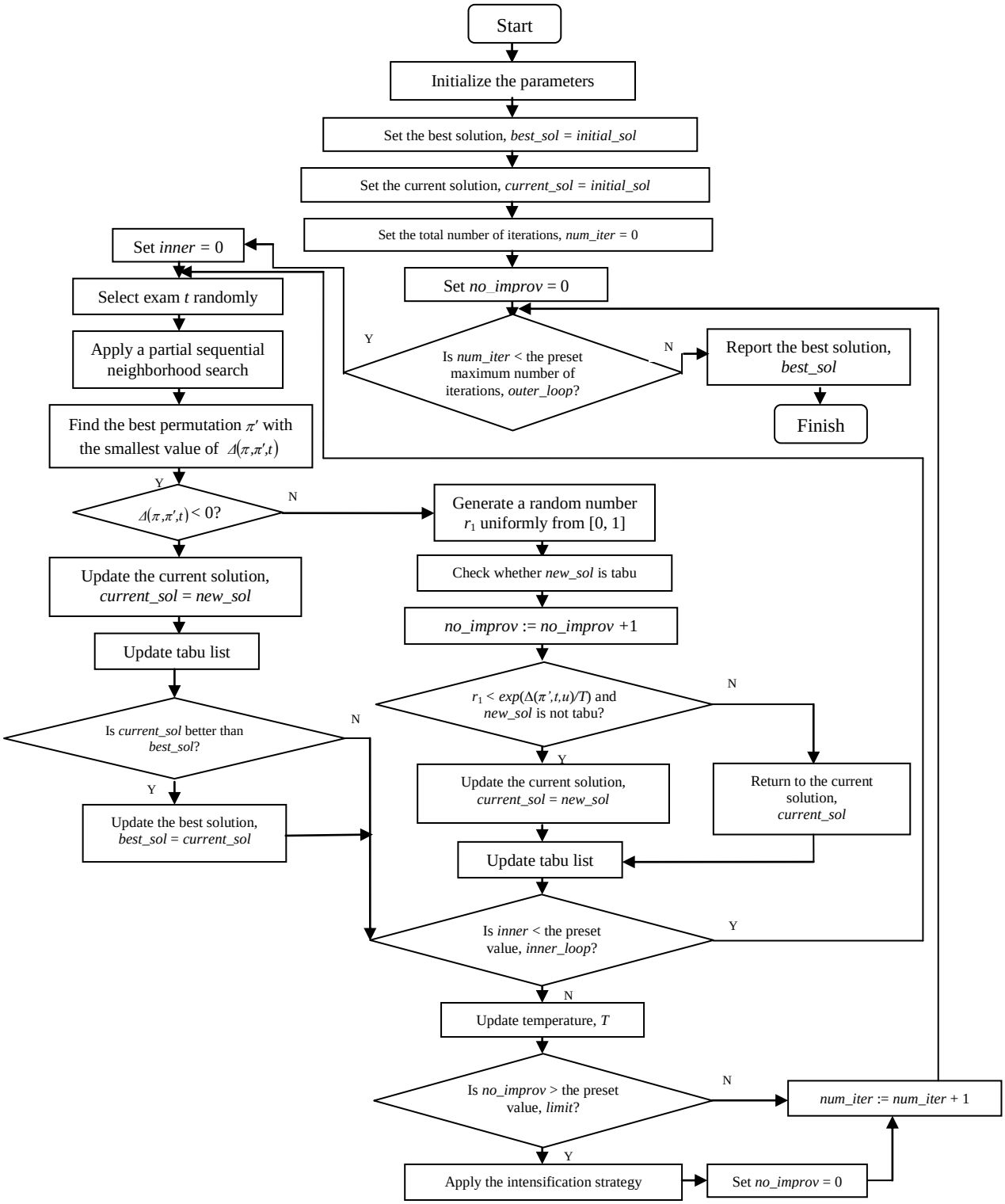
APPENDIX D2 Flow Chart of Algorithm SA-TS for the basic Examination Timetabling Problem



APPENDIX D3 Flow Chart of the Construction Phase of Modified GRASP Algorithm



APPENDIX D4 Flow Chart of Algorithm SA-TS for the Extended Examination Timetabling Problem



APPENDIX E SOLUTION REPRESENTATION

Solution for Data set 10×20(1) by Algorithm LR-SA

Day	1								2								3							
Period	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Teacher1															4(2)	4(2)			4(3)	4(3)				
Teacher2						16(1)	16(1)				17(2)	17(2)							15(1)	15(1)				
Teacher3							8(2)	8(2)							3(2)	3(2)							12(1)	12(1)
Teacher4																						2(2)	2(2)	2(2)
Teacher5									11(2)	11(2)					1(1)	1(1)			19(1)	19(1)	19(1)		7(2)	7(2)
Teacher6									13(2)	13(2)										1(2)	1(2)			
Teacher7			17(1)*	17(1)	14(3)	14(3)									14(1)	14(1)							14(2)	14(2)
Teacher8						20(2)	20(2)	20(2)					18(3)	18(3)								10(1)	10(1)	
Teacher9							6(2)	6(2)		20(1)	20(1)	20(1)									5(2)	5(2)		
Teacher10		7(3)	7(3)												9(1)	9(1)							8(1)	8(1)

Day	4								5							
Period	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Teacher1														2(1)	2(1)	2(1)
Teacher2					5(1)	5(1)			16(2)	16(2)			15(3)	15(3)		
Teacher3		12(3)	12(3)							10(2)	10(2)					
Teacher4			6(1)	6(1)								13(1)	13(1)			
Teacher5					11(1)	11(1)				7(1)	7(1)					
Teacher6					4(1)	4(1)								19(2)	19(2)	19(2)
Teacher7		15(2)	15(2)										3(1)	3(1)		
Teacher8	18(2)	18(2)												18(1)	18(1)	
Teacher9											12(2)	12(2)				
Teacher10										9(2)	9(2)					

* Teacher 7 teaches Course 17 Section 1 on Day 1 Time Periods 3 and 4