A Hybrid Algorithm for the University Course Timetabling Problem

Aldy Gunawan[†], Ng Kien Ming and Poh Kim Leng
Department of Industrial and Systems Engineering, National University of Singapore
10 Kent Ridge Crescent, Singapore 119260
aldygunawan@nus.edu.sg isenkm@nus.edu.sg isepohkl@nus.edu.sg

Abstract In this paper, we introduce a new mathematical programming model that combines teacher assignment and course scheduling problems simultaneously. Due to the limitation of a mathematical programming approach to solve large problem instances, we propose a hybrid algorithm that combines two well known metaheuristics, Simulated Annealing (SA) and Tabu Search (TS). In the proposed algorithm, useful features of each metaheuristic are exploited to obtain better solutions. Several randomly generated problem instances are used to evaluate the performance of the proposed algorithm. The computational results illustrate the ability of the hybrid algorithm to provide good quality solutions to the problem instances within reasonable computation time.

Key Words: Timetabling problem; hybrid algorithm; Simulated Annealing; Tabu Search

1. INTRODUCTION

The timetabling problem is an important and practical problem that is faced by many schools and universities. This problem has been the subject of extensive research efforts due to its wide applicability. It includes a wide range of scheduling problems: course and examination timetabling problems (Carter and Laporte 1998). 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. Two significant developments that stirred the interest in this problem are (Johnson 1993):

- Changes in the courses offered, facility requirements, number of students and teachers involved.
- Development in the computing facilities in the education institutions.

However, many research papers that tackle this problem only focus on one of the sub-problems. For example, it is often assumed that the teacher assignment problem has been solved earlier before solving the course scheduling problem, as in the works of Al-Yakoob and Sherali (2006, 2007).

In this paper, we introduce a problem that is a combination of teacher assignment and course scheduling problems at the university level, and we call it the Teacher Assignment-Course Scheduling problem (TACS problem). We also formulate a mathematical programming model that considers both problems simultaneously. The problem characteristics that we address have arisen in the context of a university in Indonesia. Although timetabling problems can vary extensively in different universities depending on their specific requirements and conditions, a number of commonly encountered requirements would be considered in the model.

[†] Corresponding author

A variety of algorithms has already been proposed to solve timetabling problems, including graph colouring algorithms (Burke et al. 1998), integer programming approaches (Al-Yakoob and Sherali 2006, 2007; Daskalaki et al. 2004), and heuristics (Aubin and Ferland 1989; Caramia et al. 2001). Over the last few years, metaheuristics have proven to be highly useful for approximately solving timetabling problems in practice. Simulated Annealing (Abramson 1991; Elmohamed et al. 1998; Bai et al. 2006) and Tabu Search (Burke et al. 2003; Costa 1994; Valdes et al. 2002; White et al.2007) are examples of metaheuristics applied to the timetabling problem.

Various combinations of algorithms or metaheuristics have also been reported in the literature for solving difficult optimization problems. This recent area of research has become more important and viable due to increasing computational power. Instead of applying only a single metaheuristic, researchers attempt to exploit and combine the advantages of various individual metaheuristics (Raidl 2006). For instance, it may be desirable to have a memory element in the Simulated Annealing approach by incorporating the Tabu Search algorithm. These combinations of metaheuristics are commonly referred to as hybrid metaheuristics.

Several different classifications of hybrid metaheuristics are presented by Blum et al. (2005), Puchinger and Raidl (2005) and Talbi (2002). There are also several research works describing applications of hybrid metaheuristics in the timetabling problems, including course and examination timetabling problems. An example is the hybrid multi-objective evolutionary algorithm proposed by Côté et al. (2005) to tackle the uncapacitated examination proximity problem. Local search operators, such as the simplified variable neighborhood descent, were implemented in order to improve the proximity cost. A method based on the Tabu Search and the Variable Neighborhood Search for solving a teacher/class timetabling problem was proposed by Kochetov et al. (2006). Computational results for randomly generated test instances show high efficiency in the proposed approach.

Some of the latest papers that apply the idea of hybrid algorithms in the university course timetabling problem are as follows: Chiarandini et al. (2006) tackled the problem by means of an algorithm based mainly on a framework consisting of the successive application of construction heuristics, variable neighborhood descent and Simulated Annealing. Rahoual and Saad (2007) applied a hybrid of two metaheuristics, Genetic Algorithm and Tabu Search, to the timetabling problem of the University of Science and Technology Houari Boumediene (USTHB). We also consider a hybrid of two metaheuristics to solve the TACS problem in this paper.

2. PROBLEM DESCRIPTION

The TACS problem considered in this paper is an extension of the basic model presented in Gunawan et al. (2006). The basic model only considers the following situations: each course can only be taught by one teacher and conducted only once a week. On the other hand, the TACS problem considers several different situations, such as:

- Some courses are divided into a number of sections due to capacity constraints and the number of students registered.
- Each course can be taught by more than one teacher.

- Time periods are considered on a day-hour basis and courses can be conducted at any time period in order to increase teaching flexibility.

The details of the TACS problem can be described as follows: a number of courses and course sections have to be allocated to teachers based on the teachers' course preferences. These allocations would be further scheduled into time periods based on the teachers' time preferences (see Figure 1 for illustration).

Although each university has its unique timetabling requirements, the following description summarizes some common requirements that are related to both teacher assignment and course scheduling problems. These requirements would be considered in the proposed mathematical programming model and treated as hard constraints that cannot be violated.

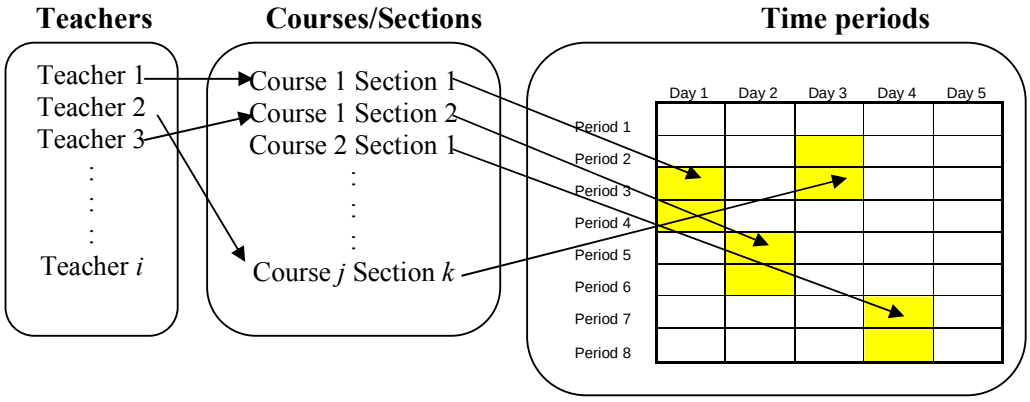

Figure 1 The TACS problem

It is required that each teacher teaches at least one course and cannot teach more than a certain number of courses. This requirement will minimize the amount of teaching preparations in terms of the number of courses assigned. Teachers will not be assigned courses that they are unable to teach. For each course, the number of teachers who can teach it is limited within certain bounds, which depend on the number of sections offered for each course. Courses with few number of sections compared with those with more number of sections would have fewer teachers to teach. Each course section is strictly taught by only one teacher. All these requirements mentioned here correspond to the teacher assignment problem.

The following describes the requirements of another sub-problem, the course scheduling problem. No teacher can be asked to teach in more than one course section at any time period. All the course sections taught by a teacher have to be spread evenly throughout a week in order to avoid unbalanced teaching load. For each course, only one section can be conducted in every time period. This requirement further ensures that students will have more opportunities to select courses.

The number of course sections taught cannot exceed the number of classrooms available during each time period. Each course section also requires a certain number of time periods to be scheduled consecutively. Only one section can be conducted each day so that all sections can be spread evenly throughout the week, except when the number of sections for a particular course is more than the number of days in a week. Finally, all courses and their planned sections must appear in the timetable.

3. THE PROPOSED MATHEMATICAL PROGRAMMING MODEL

Let I, J, and K denote the set of teachers, courses, and course sections, respectively. Every teacher $i \in I$ will teach certain course sections based on their course preference list J_i, where $J_i \subseteq J$. We also define K_j to be the set of sections of course j. It is required that each teacher i teaches not more than N_i courses.

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 paper, each time period m is assumed to be of the same duration. Each section k of course j ($k \in K_j$) 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}. For simplicity, we assume that C_{lm} is a constant, i.e., $C_{lm} = C$ for all l and all m, where C is a positive integer. Other required data parameters are listed below:

PC_{ij} value given by teacher i on the preference of being assigned to teach course j ($i \in I, j \in J$)

PT_{ilm} value given by teacher i on the preference of being assigned to teach in day l and time period m ($i \in I, l \in L, m \in M$)

LT_j minimum number of teachers who could teach course j ($j \in J$)

UT_j maximum number of teachers who could teach course j ($j \in J$)

S_j number of sections of course j ($j \in J$)

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 ($i \in I, j \in J, k \in K_j, l \in L, m \in M$)

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

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

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

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

We also let V_i denote the number of course sections taught by teacher i ($i \in I$) per day, obtained after rounding upwards to the nearest integer.

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

[*TACS*]

Maximize $$\sum_{i \in I} \sum_{j \in J} PC_{ij} P_{ij} + \sum_{i \in I} \sum_{j \in J} \sum_{k \in K_j} \sum_{l \in L} \sum_{m \in M} PT_{ilm} X_{ijklm} \qquad (1)$$

subject to:

$$\sum_{i \in I} \sum_{k \in K_j} X_{ijklm} \le 1 \qquad (j \in J, l \in L, m \in M) \qquad (2)$$

$$P_{ij} = \left\lceil \sum_{k \in K_j} \sum_{l \in L} Y_{ijkl} \Big/ S_j \right\rceil \qquad (i \in I, j \in J) \qquad (3)^*$$

---

* $\lceil a \rceil$ denotes the smallest integer greater than or equal to a

$$1 \le \sum_{j \in J} P_{ij} \le N_i \qquad (i \in I) \qquad (4)$$

$$LT_j \le \sum_{i \in I} P_{ij} \le UT_j \qquad (j \in J) \qquad (5)$$

$$\sum_{m \in M} X_{ijklm} = Y_{ijkl} H_j \qquad (i \in I, j \in J, k \in K_j, l \in L) \qquad (6)$$

$$\sum_{i \in I} \sum_{k \in K_j} Y_{ijkl} \le 1 \qquad (j \in J, l \in L) \qquad (7)$$

$$\sum_{j \in J} \sum_{k \in K_j} X_{ijklm} \le 1 \qquad (i \in I, l \in L, m \in M) \qquad (8)$$

$$\sum_{i \in I} \sum_{j \in J} \sum_{k \in K_j} X_{ijklm} \le C \qquad (l \in L, m \in M) \qquad (9)$$

$$\sum_{i \in I} \sum_{k \in K_j} \sum_{l \in L} Y_{ijkl} = S_j \qquad (j \in J) \qquad (10)$$

$$\sum_{i \in I} \sum_{l \in L} Y_{ijkl} = 1 \qquad (j \in J, k \in K_j) \qquad (11)$$

$$X_{ijklm} = 0 \qquad (i \in I, j \notin J_i, k \in K_j, l \in L) \qquad (12)$$

$$\sum_{j \in J} \sum_{k \in K_j} \sum_{l \in L} Y_{ijkl} = L_i \qquad (i \in I) \qquad (13)$$

$$V_i = \left\lceil \frac{L_i}{|L|} \right\rceil \qquad (i \in I) \qquad (14)$$

$$\sum_{j \in J} \sum_{k \in K_j} Y_{ijkl} \le V_i \qquad (i \in I, l \in L) \qquad (15)$$

$$\sum_{t=0}^{(H_j - 1)} X_{ijkl(m+t)} \ge H_j U_{ijklm} \qquad (i \in I, j \in J, k \in K_j, l \in L, m \in \{1, ..., |M| - H_j + 1\}) \qquad (16)$$

$$\sum_{i \in I} \sum_{l \in L} \sum_{m=1}^{(|M| - H_j + 1)} U_{ijklm} = 1 \qquad (j \in J, k \in K_j) \qquad (17)$$

$$\sum_{i \in I} \sum_{l \in L} \sum_{m \in M} X_{ijklm} = H_j \qquad (j \in J, k \in K_j) \qquad (18)$$

$$U_{ijklm} = 0 \qquad (i \in I, j \in J, k \in K_j, l \in L, m \in \{|M| - H_j + 2, ..., |M|\}) \qquad (19)$$

$$U_{ijklm} = 0 \qquad (i \in I, j \notin J_i, k \in K_j, l \in L, m \in \{1, ..., |M| - H_j + 1\}) \qquad (20)$$

$$X_{ijklm} \in \{0,1\} \qquad (i \in I, j \in J, k \in K_j, l \in L, m \in M) \qquad (21)$$

$$Y_{ijkl} \in \{0,1\} \qquad (i \in I, j \in J, k \in K_j, l \in L) \qquad (22)$$

$$U_{ijklm} \in \{0,1\} \qquad (i \in I, j \in J, k \in K_j, l \in L, m \in M) \qquad (23)$$

$$P_{ij} \in \{0,1\} \qquad (i \in I, j \in J) \qquad (24)$$

$$L_i \in Z^+ \qquad (i \in I) \qquad (25)$$

$$V_i \in Z^+ \qquad (i \in I) \qquad (26)$$

Equation (1) represents the objective function to be maximized, which consists of the sum of two terms: the first term refers to the total preference value of assigning courses to teachers, while the second term refers to the total preference value of assigning days and time periods to teachers to teach.

Equation (2) ensures that at most one section can be taught in every time period for a particular course *j*. Equation (3) 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 (4) restricts the number of courses that can be taught by a teacher. In this model, it is assumed that each teacher has to teach at least one course. Equation (5) restricts for each course the number of teachers who could teach it.

The relationship between variables Y_{ijkl} and X_{ijklm} is shown by equation (6). 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 (7) ensures that for any course j, at most one section can be conducted each day. Equation (8) ensures that each teacher can only be assigned at most one course section at any time period. Equation (9) prevents the total number of course sections conducted per time period from exceeding the number of classrooms available, C. Equation (10) states that all sections for each course must be scheduled in the timetable. Equation (11) ensures that each course section can only be taught by one teacher, while equation (12) ensures that teachers will not be assigned courses that they are unable to teach.

Equation (13) calculates the number of course sections taught by each teacher and equation (14) determines the number of course sections taught per day for each teacher, rounded upwards. Equation (15) helps to spread evenly all the course sections taught by each teacher throughout a week.

Equation (16) expresses the requirement 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 should be assigned to the same course section. Equation (17) ensures that there is only one starting time period for each course section, and equation (18) further ensures that the number of time periods allocated to each course section meets its requirement.

Additional constraints (19) and (20) for variables U_{ijklm} 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 also that teachers will not be assigned certain time periods for courses that they are unable to teach.

Finally, constraints (21), (22), (23) and (24) impose the 0-1 restrictions for the decision variables X_{ijkl}, Y_{ijkl}, U_{ijklm} and P_{ij}, while constraints (25) and (26) represent the nonnegative integer value requirement for the L_i and V_i variables.

Even though equations (3) and (14) are nonlinear, they can be rewritten as a linear $[TACS']$ model by introducing additional constraints (27) and (28) as follows:

$$S_j\left(\varepsilon + P_{ij} - 1\right) \le \sum_{k \in K_j} \sum_{l \in L} Y_{ijkl} \le P_{ij} S_j \qquad (i \in I, j \in J) \qquad (27)$$

$$\left|L\right|\left(\varepsilon + V_i - 1\right) \le L_i \le \left|L\right| V_i \qquad (i \in I) \qquad (28)$$

Here, ε is a small positive number such that $\varepsilon < \min\left\{\min_j\left\{1/S_j\right\}, 1/\left|L\right|\right\}$. Thus, the entire model can also be represented as follows:

[*TACS'*]:

Maximize *Objective function* (1)

subject to:

Constraints (2), (4) – (13), (15) – (28)

4. THE HYBRID ALGORITHM

Despite the linearity of the $[TACS']$ model, there is difficulty in obtaining solutions to larger instances due to the presence of integer variables. As such, a hybrid algorithm is proposed which comprises of three main components or phases: (1) pre-processing phase, (2) finding an initial feasible solution (construction phase), and (3) modifying the initial solution (improvement phase). The purpose of the first 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 $LM_i = \left\{\left(l^i, m^i\right)^1, \left(l^i, m^i\right)^2, ..., \left(l^i, m^i\right)^{|L|\times|M|}\right\}$ is the set of time periods of teacher *i* which are sorted in non-increasing order of the PT_{ilm} value.

The second phase focuses on building an initial feasible solution. We divide the TACS problem into two interrelated sub-problems: teacher assignment and course scheduling problems. The first sub-problem can be formulated as another mathematical programming model presented below with the following decision variables:

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

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

[*TA*]:

Maximize $\sum_{i\in I}\sum_{j\in J} PC_{ij} P'_{ij}$ (29)

subject to:

$$P'_{ij} = \left\lceil \sum_{k\in K_j} X'_{ijk} \Big/ S_j \right\rceil \qquad (i \in I, j \in J) \qquad (30)$$

$$1 \le \sum_{j\in J} P'_{ij} \le N_i \qquad (i \in I) \qquad (31)$$

$$LT_j \le \sum_{i\in I} P'_{ij} \le UT_j \qquad (j \in J) \qquad (32)$$

$$X'_{ijk} = 0 \qquad (i \in I, j \notin J_i, k \in K_j) \qquad (33)$$

$$\sum_{i\in I} X'_{ijk} = 1 \qquad (j \in J, k \in K_j) \qquad (34)$$

$$X'_{ijk} \in \{0,1\} \qquad (i \in I, j \in J, k \in K_j) \qquad (35)$$

$$P'_{ij} \in \{0,1\} \qquad (i \in I, j \in J) \qquad (36)$$

In the [*TA*] model, the objective function (29) only involves the course preference function. Equation (30) 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*. Equation (31) limits the number of courses taught by each teacher. Equation (32) restricts for each course the number of teachers who could teach it. Equation (33) ensures that teachers will not be assigned courses that they are unable to teach. Equation (34) assumes that each course section can only be taught by one teacher. Finally, constraints (35) and (36) represent the integrality constraints for the decision variables X'_{ijk} and P'_{ij}. The optimal solution of [*TA*]

model is denoted as initial_ta. Note that equation (30) is nonlinear and it can also be linearized by the following equation with sufficiently small but positive ε:

$$S_j\left(\varepsilon + P'_{ij} - 1\right) \le \sum_{k \in K_j} X'_{ijk} \le S_j P'_{ij} \qquad \left(i \in I, j \in J\right) \tag{37}$$

The second phase continues with solving the second sub-problem, the course scheduling problem. To build an initial feasible solution to this sub-problem, we propose a simple greedy heuristic which is similar to the heuristic in Gunawan et al. (2007a). A feasible initial solution, initial_cs, is first constructed by satisfying as much course and time period preferences as possible. For each course j, a teacher with the highest course preference is selected from the list I_j. This course j is then scheduled to day l and time period m with the highest time period preference by taking list LM_i into consideration.

In the third phase, the improvement phase, we provide a framework involving the hybridization of Simulated Annealing (SA) and Tabu Search to develop and improve the quality of the solution. The proposed hybrid algorithm is mainly based on Simulated Annealing (Kirkpatrick 1983). The main difference of the standard SA and the proposed SA lies in the additional elements or strategies introduced. Several Tabu Search features, such as tabu length, tabu list and the intensification strategy are embedded in the algorithm for further improvement (Glover 1989).

The phase is started by applying two operations, the re-allocation of teachers to courses and course sections, followed by rescheduling these changes into days and time periods. The first operation is started by randomly choosing course j that is currently taught by teacher i_1, followed by finding another new teacher $i_2 \neq i_1 \left(i_2 \in I_j\right)$ without violating the maximum load N_{i_2}. Two possible moves will be taken into consideration as shown in Figure 2.

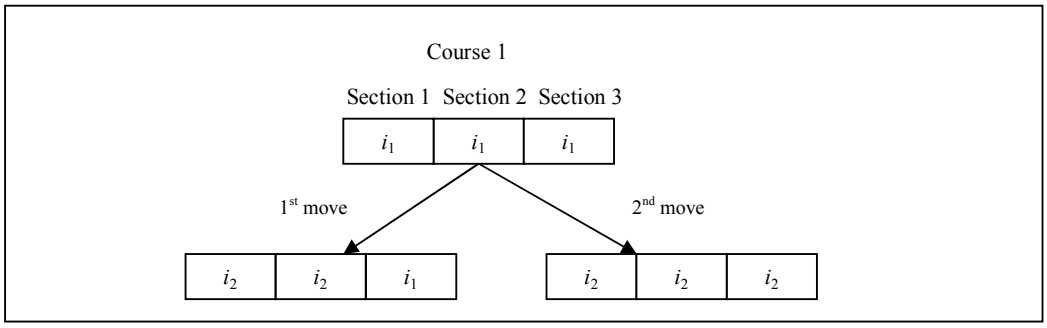

Figure 2 Two possible moves

We 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 (first possible move), or teacher i_2 will fully replace teacher i_1 on course j (second possible 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 last alternative. Once this operation is completed, the change of objective function Δ will then be evaluated by Algorithm 1 (Figure 3).

Algorithm 1:
ACCEPT-REJECT 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) Update tabu list
(8) else
(9) Choose a random number r_1 uniformly from [0,1]
(10) Check whether the new solution is taboo
(11) **If** $r_1 < e^{-\Delta / T_n}$ and the new solution is not taboo
(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 3 Evaluation process of SA

As described earlier, several features from Tabu Search are incorporated to further evaluate the modification. The new allocation would be accepted if it can provide a better allocation than the previous one. This admissible move can be either a non-tabu or a tabu move which passes the aspiration level criterion.

In the standard Simulated Annealing algorithm, a deteriorating move would be evaluated by using a probabilistic acceptance criterion:

$$P(\text{Acceptance} / \Delta, T_n) = e^{-\Delta / T_n}, \tag{38}$$

where T_n is the temperature at iteration n. In an effort to avoid excessive or unnecessary moves which will deteriorate the objective function value especially during high temperatures, we add an additional evaluation step after the probabilistic acceptance calculation. When a move belongs to the tabu list for a given iteration, it is not allowed to be accepted, i.e., only a non-tabu move can be accepted. Finally, the tabu list is updated. The tabu list in the first operation is denoted as `tabu1`, which contains pairs of teacher i and course j visited in the last `length1` iterations.

The improvement phase is then continued to the second operation. Let K_{i_2} be the set of sections of course j taken over by teacher i_2. Suppose the teacher reallocation is accepted. 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 also introduce another tabu list for this operation (`tabu2`), which contains a list of {teacher i, course j, section k, day l, and starting time period m} visited in the last `length2` iterations. However if the first operation is rejected, the second operation is still conducted 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 also continued by evaluating the objective function value and checking the tabu list, `tabu3`. In this operation, the tabu list

only contains a list of {course j, day l, and starting time period m}, which is forbidden in the last `length3` iterations.

An intensification strategy is also applied in the proposed algorithm. This strategy focuses the search once again starting from the best solution obtained if we cannot improve the solution obtained so far within a certain number of iterations (`limit`). Finally, the entire algorithm will be terminated if the total number of iterations of the outer loop reaches the preset maximum number of iterations, `max_count`.

The details of the proposed algorithm are presented in Figure 4.

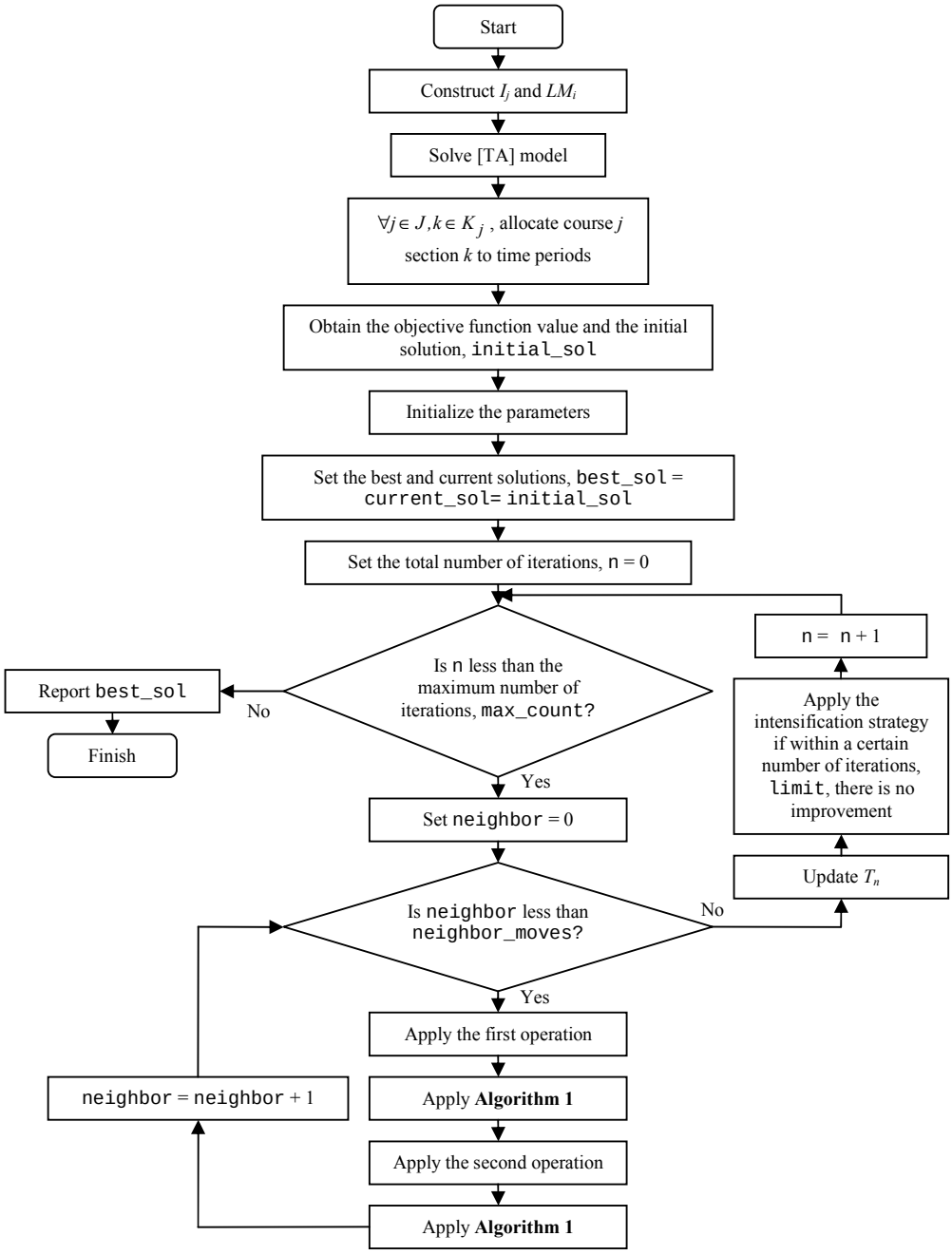

Figure 4 Flowchart of the proposed algorithm

5. COMPUTATIONAL RESULTS

Computational experiments to evaluate the performance of the proposed algorithm were performed on two different groups of randomly generated data sets with distinct characteristics. Tables I and II summarize the characteristics of each data set. The main difference between these two groups lies in the number of teachers $|I|$ and number of courses $|J|$. The algorithm was implemented using C++ on a 2.6GHz Intel Pentium 4 with 512 MB of RAM under the Microsoft Windows XP Operating System. The optimal solutions of [*TACS'*] and [*TA*] models were obtained by using ILOG OPL Studio 4.2 with the same operating system.

Table I
CHARACTERISTICS OF GROUP I DATA SETS

Data set	Number of teachers	Number of courses	Number of sections	Number of days	Number of time periods per day	Maximum load per teacher	Number of classrooms available
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 II
CHARACTERISTICS OF GROUP II DATA SETS

Data set	Number of teachers	Number of courses	Minimum number of sections	Maximum number of sections	Number of days	Number of time periods per day	Maximum load per teacher	Number of classrooms available
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

The values of the parameters used in the computational study are summarized in Table III. These values are determined experimentally to ensure a compromise between computation time and solution quality.

The [*TACS'*] model was initially solved by 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, we only report the best known solutions that could be obtained within 24 hours for those data sets. 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.

For each data set, the proposed algorithm was executed 20 times with different random seeds. Table IV summarizes the overall results that include the average objective function value obtained, the best objective function value obtained and the average CPU time required to obtain the solution (in seconds). The results obtained were compared with the best known/optimal solutions generated by ILOG OPL Studio 4.2 and those of earlier work (Gunawan et al. 2007b), which only applied the idea of Simulated Annealing (Algorithm SA1).

Table III
PARAMETER SETTINGS FOR HYBRID ALGORITHM

Parameter	Value
Number of iterations, `max_count`	$\|I\|\|L\|\|M\|$
Initial temperature, T_0	10,000
Number of neighborhood moves at each temperature T_n, `neighbor_moves`	$\|I\|\|L\|\|M\|$
Cooling factor α	0.95
Number of non-improvement iterations prior to intensification, `limit`	$0.05\|I\|\|L\|\|M\|$
Length of `tabu1`, `length1`	$0.25\|I\|$ for Group I data sets $0.5\|I\|$ for Group II data sets
Length of `tabu2`, `length2`	$\|L\|$ for Group I data sets $2\|L\|$ for Group II data sets
Length of `tabu3`, `length3`	$\|L\|$ for Group I data sets $2\|L\|$ for Group II data sets

Table IV
COMPUTATIONAL RESULTS OF PROPOSED HYBRID ALGORITHM AND OTHER SOLUTION APPROACHES

Data set	Solution obtained by commercial software		Algorithm SA1			Hybrid Algorithm		
	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	980	1.82	930	930	0.09	980	980	0.77
5×5_2	1,210	2.25	1,130	1,130	0.08	1,210	1,210	0.71
10×10_1	2,200	22.15	2,020	2,020	1.59	2,200	2,200	5.91
10×10_2	2,780	19.70	2,560	2,560	1.48	2,777.5	2,780	4.39
15×15_1	3,270	200.32	3,140	3,140	5.30	3,270	3,270	18.18
15×15_2	4,150	189.92	3,850	3,850	4.91	4,141	4,150	18.78
20×20_1	4,540	172.26	4,420	4,420	10.94	4,540	4,540	30.63
20×20_2	5,660	663.78	5,460	5,460	10.98	5,574.5	5,610	30.30
10×20_1	7,800	647.54	6,913	6,940	3.79	7,429.5	7,490	6.47
10×20_2	7,660	1,114.60	6,598	6,670	3.98	7,186	7,360	6.81
20×30_1	11,140	9,265.63	9,621	9,680	6.72	10,718.5	10,920	37.73
20×30_2	12,880	55,032.5	11,132	11,370	10.96	12,254.5	12,390	29.79
20×40_1	13,210[a]	-[b]	11,680.5	11,940	13.45	12,704	12,920	128.36
20×40_2	16,190[a]	-[b]	14,136	14,200	19.86	14,957.5	15,120	101.42
30×60_1	21,890[a]	-[b]	19,250	19,300	52.01	20,806.5	20,910	379.60
30×60_2	24,480[a]	-[b]	20,643	20,930	78.98	22,793.5	23,180	394.47

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

Based on the average objective function value, we observe that the hybrid algorithm is able to obtain better results than those obtained by Algorithm SA1. In order to compare the obtained results to the optimal solutions, we also show the best known solution for each data set. The hybrid algorithm is able to obtain the optimal solution for most of Group I data sets.

From the average CPU times, we observe that the computation time taken by algorithm SA1 is smaller than the computation time taken by the hybrid algorithm in all tests. This is due to the difference in the improvement phases of both algorithms, with the evaluation process running twice in the improvement phase of the hybrid algorithm. Nevertheless, better results on the objective function values are obtained.

Table V compares the objective function value obtained between Algorithm SA1 and the proposed hybrid algorithm. The comparison is done by calculating the deviation of the best and the average objective function values of the proposed algorithms from the best known/optimal value, denoted as Φ_1 and Φ_2:

$$\Phi_1 = 100 \times \left(\frac{\text{best known/optimal solution} - \text{average objective function value of algorithm}}{\text{best known/optimal solution}} \right)$$

$$\Phi_2 = 100 \times \left(\frac{\text{best known/optimal solution} - \text{best objective function value of algorithm}}{\text{best known/optimal solution}} \right)$$

Table V
COMPARISON OF DEVIATIONS FOR ALGORITHM SA1 AND THE HYBRID ALGORITHM

Data set	Algorithm SA1		Hybrid Algorithm			
	Φ_1 (A)	Φ_2 (B)	Φ_1 (C)	Φ_2 (D)	(A) – (C)	(B) – (D)
5×5_1	5.10	5.10	0.00	0.00	5.10	5.10
5×5_2	6.61	6.61	0.00	0.00	6.61	6.61
10×10_1	8.18	8.18	0.00	0.00	8.18	8.18
10×10_2	7.91	7.91	0.09	0.00	7.82	7.91
15×15_1	3.98	3.98	0.00	0.00	3.98	3.98
15×15_2	7.23	7.23	0.22	0.00	7.01	7.23
20×20_1	2.64	2.64	0.00	0.00	2.64	2.64
20×20_2	3.53	3.53	1.51	0.88	2.02	2.65
10×20_1	11.37	11.03	4.75	3.97	6.62	7.05
10×20_2	13.86	12.92	6.19	3.92	7.68	9.01
20×30_1	13.64	13.11	3.78	1.97	9.85	11.13
20×30_2	13.57	11.72	4.86	3.80	8.72	7.92
20×40_1	11.58	9.61	3.83	2.20	7.75	7.42
20×40_2	12.69	12.29	7.61	6.61	5.07	5.68
30×60_1	12.06	11.83	4.95	4.48	7.11	7.35
30×60_2	15.67	14.50	6.89	5.31	8.78	9.19

The proposed hybrid algorithm provides better results in terms of the quality of the solution as indicated by Φ_1 and Φ_2. We observe that the proposed hybrid algorithm yields good solutions with the values of Φ_1 and Φ_2 not exceeding 7.61% and 6.61% respectively. The hybrid algorithm outperforms SA1 in terms of the objective function values obtained. The values of Φ_1 and Φ_2 are reduced by up to 9.85% and 11.13% respectively.

6. CONCLUSIONS

The main motivation for this work is to consider two interrelated sub-problems, teacher assignment and course scheduling problems simultaneously. A new mathematical programming model for the associated timetabling problem has been proposed. In this work, we proposed a hybrid algorithm that incorporates Simulated Annealing and Tabu Search algorithms for solving the problem. Computational experiments show that the proposed algorithm is able to produce good quality solutions within reasonable computation time when compared to previous research work.

A possible future research area is to develop other methods that might solve the problem more efficiently. For example, the Tabu Search framework was designed with only short term memory and it might be useful to implement other

strategies, including long term memory and diversification strategy. Another future research area is to incorporate additional requirements that might be required by other universities and then solve the resulting problem using the hybrid algorithm. It is also possible to consider extending the hybrid algorithm to solve other types of timetabling problems.

REFERENCES

1. Abramson D (1991) Constructing school timetables using simulated annealing: sequential and parallel algorithm. Management Science 37:98-113
2. Al-Yakoob SM, Sherali HD (2007) A mixed integer programming approach to a class timetabling problem: a case study with gender policies and traffic considerations. European Journal of Operational Research 180:1028-1044
3. Al-Yakoob SM, Sherali HD (2006) Mathematical programming models and algorithms for a class-faculty assignment problem. European Journal of Operational Research 173:488-507
4. Aubin J, Ferland JA (1989) A large scale timetabling problem. Computers and Operations Research 16:67-77
5. Bai R, Burker EK, Kendall G, Collum BM (2006) A simulated annealing hyper-heuristic for university course timetabling. In: Proc. International Conference on the Practice and Theory of Automated Timetabling VI (30 August – 1 September 2006, Brno, Czech Republic):347350
6. Blum C, Roli A, Alba E (2005) An introduction to metaheuristic techniques. In: Alba E (ed) Parallel metaheuristics, a new class of algorithms. Wiley, Hoboken:3-42
7. Burke EK, Causemacker PD, Berghe VG (2004) Applications to timetabling. In: Gross JL, Yellen J (ed) Handbook of graph theory. CRC Press, Boca Raton:445-474
8. Burke EK, Kendall G, Soubeiga, EA (2003) Tabu-search hyperheuristic for timetabling and rostering. Journal of Heuristics 9:451-470
9. Burke EK, Kingston JH, Pepper PA (1998) A standard data format for timetabling instances. In: Burke EK, Carter M (ed) The Practice and Theory of Automated Timetabling II (PATAT'97, Selected Papers). Lecture Notes in Computer Science, Vol. 1408, Springer, Berlin:213-222
10. Caramia M, Olmo PD, Italiano GF (2001) New algorithms for examination timetabling. In: Naher S, Wagner D (ed) Algorithm Engineering – Proceedings of the 4th International Workshop, WAE 2000. Lecture Notes in Computer Science, Vol. 1982, Springer, Berlin:230-241
11. Carter MW, Laporte G (1998) Recent developments in practical course timetabling. In: Burke EK, Carter M (ed) The Practice and Theory of Automated Timetabling II (PATAT'97, Selected Papers). Lecture Notes in Computer Science, Vol. 1408, Springer, Berlin:3-19
12. Chiarandini M, Socha K, Birattari M, Doria OR (2006) An effective hybrid approach for the university course timetabling problem. Journal of Scheduling 9:403-432
13. Costa D (1994) A tabu search algorithm for computing an operational timetable. European Journal of Operational Research 76:98-110
14. Côté P, Wong T, Sabourin R (2005) A hybrid multi-objective evolutionary algorithm for the uncapacitated exam proximity problem. In: Burke EK, Trick M (ed) The Practice and Theory of Automated Timetabling V (PATAT'04,

Selected Papers). Lecture Notes in Computer Science, Vol. 3616, Springer, Berlin:294-312
15. Daskalaki S, Birbas T, Housos E (2004) An integer programming formulation for a case study in university timetabling. European Journal of Operations Research 153:117-135
16. Elmohamed MAS, Coddington P, Fox G (1998) A comparison of annealing techniques for academic course scheduling. In: Burke EK, Carter M (ed) The Practice and Theory of Automated Timetabling II (PATAT'97, Selected Papers). Lecture Notes in Computer Science, Vol. 1408, Springer, Berlin:92-112
17. Glover F (1989) Tabu search – part I. ORSA Journal on Computing 1:190-206
18. Gunawan A, Ng KM, Poh KL (2006) A mathematical programming model for a timetabling problem. In: Proc. International Conference on Scientific Computing, USA
19. Gunawan A, Ng KM, Poh KL (2007a) An improvement heuristic for the timetabling problem. International Journal Computational Science 1:162-178
20. Gunawan A, Ng KM, Poh KL (2007b) Solving the teacher assignment-course scheduling problem by a hybrid Algorithm. International Journal of Computer, Information and Systems Science and Engineering 1:136-141
21. Johnson D (1993) A database approach to course timetabling. The Journal of the Operational Research Society 44:425-433
22. Kirkpatrick S, Gellatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671-680
23. Puchinger J, Raidl GR (2005) Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification. In: Mira J, Alvarez J R (ed) Arfiticial Intelligence and Knowledge Engineering Applications: a Bioinspired Approach. Lecture Notes in Computer Science, Vol. 3562, Springer, Berlin:41-53
24. Rahoual M, Saad R (2007) Solving timetabling problems by hybridizing genetic algorithms and tabu search. In: Burke EK, Rudová H (ed) The Practice and Theory of Automated Timetabling VI (PATAT'06, Selected Papers). Lecture Notes in Computer Science, Vol. 3867, Springer, Berlin:467-472
25. Raidl GR (2006) A unified view on hybrid metaheuristics. In: Almedia F, Aguilera MJB, Blum C, Vega JMM, Pérez MP, Roli A, Sampels M (ed) Hybrid Metaheuristics – Proceedings of the 3rd International Workshop, HM 2006. Lecture Notes in Computer Science, Vol. 4030, Springer, Berlin:1-12
26. Talbi EG (2002) A taxonomy of hybrid metaheuristics. Journal of Heuristics 8 :541-565
27. Valdes RA, Crespo E, Tamarit JM (2002) Design and implementation of a course scheduling system using tabu search. European Journal of Operational Research 137:512-523
28. White CA, Nano E, Ngoc DHN, White GM (2007) An evaluation of certain heuristic optimization algorithms in scheduling medical doctors and medical students. In: Burke EK, Rudová H (ed) The Practice and Theory of Automated Timetabling VI (PATAT'06, Selected Papers). Lecture Notes in Computer Science, Vol. 3867, Springer, Berlin:320-328