International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:1, No:9, 2007

Solving the Teacher Assignment-Course Scheduling Problem by a Hybrid Algorithm

Aldy Gunawan, Kien Ming Ng, and Kim Leng Poh

***Abstract*—**This paper presents a hybrid algorithm for solving a timetabling problem, which is commonly encountered in many universities. The problem combines both teacher assignment and course scheduling problems simultaneously, and is presented as a mathematical programming model. However, this problem becomes intractable and it is unlikely that a proven optimal solution can be obtained by an integer programming approach, especially for large problem instances. A hybrid algorithm that combines an integer programming approach, a greedy heuristic and a modified simulated annealing algorithm collaboratively is proposed to solve the problem. Several randomly generated data sets of sizes comparable to that of an institution in Indonesia are solved using the proposed algorithm. Computational results indicate that the algorithm can overcome difficulties of large problem sizes encountered in previous related works.

***Keywords*—**Timetabling problem, mathematical programming model, hybrid algorithm, simulated annealing.

I. Introduction

TIMETABLING problems arise in a wide variety of fields including education, transportation, sports, and healthcare institutions. It is well known that the timetabling problem is NP-complete [1]. Here, we focus on a special class of timetabling problems, known as the university course timetabling problem. This problem is commonly encountered in many universities throughout the world. It can be further classified into five different sub-problems: teacher assignment, course scheduling, class-teacher timetabling, student scheduling, and classroom assignment [2].

This paper deals with the first two sub-problems simultaneously. The problem faced in teacher assignment is how to assign and schedule the teachers to the courses and course sections by taking some factors, such as teachers' preferences and the number of courses offered, into consideration. The timetabling process will then be continued with scheduling course sections to time periods, which is known as the course scheduling problem. From the literature, we notice that most of the papers only focus on one of the sub-problems. For example, papers about the course scheduling problem often assume that the teacher assignment problem has been solved earlier before solving the course scheduling problem.

Many approaches have been proposed for solving timetabling problems, such as exact algorithms and heuristics. The emphasis of this paper is to develop a hybrid algorithm to solve the problem. Another contribution of this paper is to consider both the teacher assignment and course scheduling problems simultaneously. This problem becomes more complex than if teaching assignment and course scheduling problems are considered separately.

The rest of the paper is organized as follows: Section II gives related works of the timetabling problem. A detailed description of the timetabling problem is presented in Section III. Section IV briefly describes a proposed mathematical programming model for the problem. A proposed hybrid algorithm is then described in Section V, and the results of the computational experiments are reported and discussed in Section VI. Finally, an overall conclusion and suggestions for further research work are given in Section VII.

II. Related Works

The timetabling problem is one of the scheduling problems that has been extensively studied and published in Operations Research literature over the last 25 years [3]. The solution approaches range from graph coloring to heuristic algorithms, including mathematical programming models and metaheuristics as well.

For many years, the main focus of research in the timetabling problem was on the application of a single solution approach. A large variety of such approaches have been tried out, such as an integer programming approach [4], tabu search [3], and simulated annealing [5]. Recently, some researchers have attempted to combine several approaches, such as hybridization of exact algorithms and metaheuristics. Hybrid algorithms exploit the strength of different methods by applying them to problems that can be solved efficiently.

The hybridization of exact algorithms and metaheuristics can be categorized into two classes, commonly known as collaborative and integrative combinations [6]. Collaborative combination refers to the algorithms that are executed sequentially, intertwined or in parallel. These algorithms exchange information, but are not part of each other. In

Manuscript received June 15, 2007.

Aldy Gunawan is with the Department of Industrial and Systems Engineering, National University of Singapore, 1 Engineering Drive 2, S(117576), Singapore (phone: 65-6516-2008; fax: 65-6516-1434; e-mail: aldygunawan@nus.edu.sg).

Kien Ming Ng is with the Department of Industrial and Systems Engineering, National University of Singapore, 1 Engineering Drive 2, S(117576), Singapore (e-mail: isenkm@nus.edu.sg).

Kim Leng Poh is with the Department of Industrial and Systems Engineering, National University of Singapore, 1 Engineering Drive 2, S(117576), Singapore (e-mail: isepohkl@nus.edu.sg).

integrative combinations, one algorithm is a subordinate embedded component of another algorithm.

One of the earlier related papers in the university course timetabling problem that has applied the idea of hybrid algorithms was presented by Weare *et al*. [7]. This paper proposed a hybrid genetic algorithm that combines a direct representation of the timetable and heuristic crossover operators. Petrovic and Yang [8] presented a combination of case-based reasoning system and the Great Deluge Algorithm for solving examination timetabling problems. Some other applications of the hybrid algorithm in timetabling problems were also presented by Chiarandini and Stützle [9], Duong and Lam [10] and Merlot *et al.* [11].

One of the latest applications of the hybrid algorithm for timetabling problem was presented by Chiarandini *et al*. [12]. It describes a hybrid metaheuristic algorithm for solving the university course timetabling problem. The entire framework that consists of the successive application of several heuristics is promising in achieving good results.

III. Problem Description

The timetabling problem that we address has arisen in the context of a university in Indonesia. In every new semester, several courses are offered to students. Each course can be divided into different sections due to the classroom capacity constraint and the number of students registered. Teachers are allowed to choose the courses that they are willing to teach based on their preferences, along with their preferred days and time periods.

As mentioned earlier, the primary problem discussed in this paper is the combination of two sub-problems: teacher assignment and course scheduling problems. Instead of solving these sub-problems separately, we focus on how to solve them simultaneously by taking the requirements of both sub-problems into consideration.

Some efforts in this area of research have already been started by using the mathematical programming approach [13, 14], in which it is shown that timetabling problems with data sizes comparable to that of an institution can be solved with the help of the models. However, it is found that mathematical programming models were not an effective way for finding the existence of an optimal solution, especially for large-scale timetabling problems. Thus, the design of heuristic approaches was proposed.

Although each university has some unique combination of requirements, the most common forms of requirements that might also be encountered in other universities would be accommodated in the model. The requirements imposed are as follows:

a) For each course, only one section can be conducted in every time period.
b) Each teacher has to teach at least one course and cannot teach more than a certain number of courses.
c) The number of teachers who can teach each course is limited.
d) All course sections have to be spread evenly throughout a week, so that for a particular course, only one section can be conducted every day.
e) Each teacher can only teach at most one course section in a particular time period.
f) The number of course sections taught cannot exceed the number of classrooms available during each time period.
g) All sections for a particular course must be scheduled.
h) Each course section can only be taught by one teacher.
i) Each teacher will not be assigned courses that he/she is unable to teach.
j) All the course sections taught by a teacher will be spread out evenly during a week.
k) Each course section has to be scheduled in a certain number of time periods consecutively.

The above requirements would be accommodated in the proposed mathematical programming model and regarded as hard constraints that cannot be violated.

IV. Mathematical Programming Model

In order to compare the solutions obtained by an integer programming approach and proposed hybrid algorithms, the problem is formulated as a mathematical programming model (model [*A*]). Let I, J, and K be the set of teachers, courses, and course sections, respectively. Every teacher i will teach some course sections based on their course preference list J_i, where $J_i \subset J$ and he/she must not teach more than N_i courses. PC_{ij} is the value given by teacher i on the preference to be assigned to teach course j. The number of teachers teaching course j is also bounded by the minimum and maximum values, LT_j and UT_j. The set of sections of course j is denoted by K_j, while S_j is the number of sections of course j.

The timetable is in the form of a weekly schedule with the set of days in a week being denoted by L, and the set of time periods being denoted by M. In this paper, each time period has the same duration. Each course section has to be scheduled into time periods based on the number of time periods required. The number of classrooms available per time period is limited by C_{lm}. We assume that $C_{lm} = C$ for all $l \in L$ and $m \in M$. The value given by teacher i on the preference to be assigned to teach in day l and time period m is denoted as PT_{ilm}.

Some of the decision variables defined are as follows:

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

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

In our problem, the objective function (1) below reflects a preference function that needs to be maximized. It refers to the preferences of assigning course j to teacher i and the preference of scheduling course j and section k to time period m and day l. It is assumed that these preferences are equally important.

[Model *A*]

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

Some of the main constraints encountered in our timetabling problem are depicted next and the full details of

this mathematical programming model (model [*A*]) can be found in [14].

Requirement a:

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

Requirement b:

$$1 \le \sum_{j\in J} P_{ij} \le N_i \quad (\text{all } i\in I) \tag{3}$$

Requirement c:

$$LT_j \le \sum_{i\in I} P_{ij} \le UT_j \ (\text{all } j\in J) \tag{4}$$

Requirement f:

$$\sum_{i\in I}\sum_{j\in J}\sum_{k\in K_j} X_{ijklm} \le C \ (\text{all } l\in L, \text{ all } m\in M) \tag{5}$$

$$X_{ijklm} \in \{0,1\}$$
(all $i\in I$, all $j\in J$, all $k\in K_j$, all $l\in L$, all $m\in M$) (6)

$$P_{ij} \in \{0,1\} \quad (\text{all } i\in I, \text{ all } j\in J) \tag{7}$$

V. Proposed Hybrid Algorithm

In this section, we describe the proposed hybrid algorithm for solving the problem. It combines an integer programming approach, a greedy heuristic and a simulated annealing algorithm sequentially. The proposed algorithm consists of three phases: (1) pre-processing, (2) construction, and (3) improvement.

The first phase deals with pre-processing data. Each course j has a list of teachers, I_j, who are sorted in non-increasing order of the given preferences for each teacher i being allocated to course j. The given time period preferences (day l and time period m) for each teacher i are also sorted in non-increasing order, with the list being called LM_i. The time complexity for these processes is $O(|I|^2|J|)$ and $O(|I||L|^2|M|^2)$, respectively. The details are described in Algorithm 1 as shown in Fig. 1 below.

Algorithm 1: PRE-PROCESSING PHASE ( )
(1) **For** j = 1 **to** $|J|$
(2) construct I_j
(3) **For** i = 1 **to** $|I|$
(4) construct LM_i

Fig. 1 Pseudocode of the pre-processing phase

The primary purpose of the construction phase is to build an initial feasible solution. The problem is divided into two sub-problems. The first sub-problem, which deals with the teacher assignment problem, is considered as an easy problem that can be solved optimally by an integer programming approach. The constraints related to the teacher assignment problem are taken into consideration.

The resulting mathematical programming model (Model [*B*]), which is a part of model [*A*], is proposed. We define some additional decision variables:

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

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

Model [*B*] is then formulated as follows:

$$\text{Max} \sum_{i\in I}\sum_{j\in J} PC_{ij} \times P'_{ij} \tag{8}$$

subject to

$$P'_{ij} = \left\lceil \sum_{k\in K_j} \frac{X'_{ijk}}{S_j} \right\rceil \quad (\text{all } i\in I, \text{ all } j\in J) \tag{9}$$

$$1 \le \sum_{j\in J} P'_{ij} \le N_i \quad (\text{all } i\in I) \tag{10}$$

$$LT_j \le \sum_{i\in I} P'_{ij} \le UT_j \ (\text{all } j\in J) \tag{11}$$

$$X'_{ijk} = 0 \quad (\text{all } i\in I, \text{ all } j\notin J_i, \text{ all } k\in K_j) \tag{12}$$

$$\sum_{i\in I} X'_{ijk} = 1 \quad (\text{all } j\in J, \text{ all } k\in K_j) \tag{13}$$

$$X'_{ijk} \in \{0,1\} \quad (\text{all } i\in I, \text{ all } j\in J, \text{ all } k\in K_j) \tag{14}$$

$$P'_{ij} \in \{0,1\} \quad (\text{all } i\in I, \text{ all } j\in J) \tag{15}$$

The objective function (8) reflects a course preference function that needs to be maximized. Equation (9) indicates that if teacher i teaches at least one section of course j, the value of P'_{ij} will be 1, meaning that teacher i teaches course j. Here, $\lceil a \rceil$ denotes the smallest integer greater than or equal to a. This equation involves nonlinear functions of the decision variables and these can always be linearized by adding some additional constraints. Equation (10) ensures that teachers have to teach at least one course and cannot exceed their maximum number of courses allowed. Equation (11) limits the number of teachers for each course. Equation (12) ensures that teachers will not be assigned courses that they are unable to teach. Equation (13) assumes that each course section can only be taught by one teacher. Finally, equations (14) and (15) impose the 0-1 restrictions on the decision variables X'_{ijk} and P'_{ij}.

The solution obtained from this sub-problem is then treated as the initial solution for the next sub-problem. The second sub-problem, which deals with the course scheduling problem, is difficult to solve especially when the problem size is large. Therefore, it would be solved by a simple greedy heuristic instead. The ideas of a simple greedy heuristic as well as the parameter values used are similar to that of the heuristic proposed by Gunawan *et al.* [14]. The time complexity of this phase is $O(|I||J||K||L||M|)$. The entire process in the construction phase is briefly outlined in Algorithm 2 as shown in Fig. 2.

Algorithm 2: CONSTRUCTION PHASE ( )
(1) Solve Model [*B*], determining X'_{ijk} and P'_{ij}
(2) **For** j = 1 **to** $|J|$
(3) **For** k = 1 **to** S_j
(4) Allocate course j section k to time periods
(5) Calculate the objective function value
(6) Set the solution obtained as the initial solution, `initial`

Fig. 2 Pseudocode of the construction phase

In the improvement phase, we propose a modified simulated annealing (SA) algorithm. SA was originally developed by Kirkpatrick *et al*. [15] for finding good solutions to a wide variety of combinatorial optimization problems, such as the traveling salesman problem, machine scheduling problem and timetabling problem. The SA algorithm is a type of local-search heuristic algorithm that avoids getting trapped at a local maximum by accepting "downhill" moves which decrease the objective function value using a probabilistic acceptance criterion.

The acceptance or rejection of a downhill move is determined by a random acceptance function that is equal to $exp\ (-\Delta/T)$. T is the control parameter, called temperature in analogy with the physical annealing process and Δ is the difference of objective function values between two successive moves.

The most commonly used cooling schedule, geometric cooling schedule, together with a specific type of neighborhood structure, is applied to our problem. An additional modification is also introduced in order to further improve the quality of the solutions. We apply the intensification strategy that is originally from the tabu search (TS) algorithm. Suppose there is no improvement of the solution obtained within a certain number of iterations (*LIMIT*), the solution search is focused and started from the best solution obtained so far. The following figures represent the details of the improvement phase.

Algorithm 3: MODIFIED SIMULATED ANNEALING ( )
(1) Set the temperature $T = T_0$
(2) Set the current solution, `current` = `initial`
(3) Set the temporary solution, `temp` = `initial`
(4) Set the best solution, `best` = `initial`
(5) Set the total number of iterations without improvement, `numb_iter_no_improv` = 0
(6) Set `limit` = 0
(7) **While** the total number of iterations, `numb_iter` is less than the preset maximum number of iterations, `max_count` **do**:
(8) **Repeat** `neighbor_moves` times:
(9) Apply NEIGHBORHOOD IMPROVEMENT ( )
(10) Update temperature $T_{numb\_iter+1} = \alpha \times T_{numb\_iter}$
(11) **If** (`numb_iter_no_improv` > `limit`)
/* intensification strategy */
(12) Set the current solution, `current` = `best`
(13) Set the temporary solution, `temp` = `best`
(14) Return to the best solution, `best`

Fig. 3 Pseudocode of the improvement phase

After an initial solution is obtained from the previous phase, two operations are performed in order to seek better improvements by exploring the neighborhoods of the current solution. These two operations are re-allocation of teachers to courses and course sections, followed by re-scheduling of these changes into days and time periods. The details of the neighborhood improvement are described in Algorithm 4.

Algorithm 4: NEIGHBORHOOD IMPROVEMENT ( )
(1) Choose course $j \in J$ randomly
(2) Set `old_teacher` = the teacher who teaches course j, if there is more than one teacher, choose one randomly
(3) Find another teacher $i \in I_j$ (`new_teacher`) for being allocated to course j
(4) **If** `new_teacher` does exist:
/* teacher replacement is feasible */
(5) Choose a random number r_1 uniformly from [0, 1]
(6) **If** ($r_1 < 0.5$) AND (the number of sections of course j taught by `old_teacher` > 1) AND (the number of teachers allocated to course j < the maximum allowed):
(7) Choose the number of sections replaced by `new_teacher` randomly, `numb`
(8) Replace `old_teacher` with `new_teacher` for selected sections
(9) **Else** ***/* replacement process */***
(10) set `numb` = the total number of sections of course j taught by `old_teacher`
(11) Replace `old_teacher` with `new_teacher` for `numb` sections
(12) **For** $a = 1$ to `numb`
(13) Set `old_period` = the current time (day and time periods) of course j section a
(14) Check feasibility if `new_teacher` is allocated to `old_period`
(15) **If** feasible
(16) Allocate `new_teacher` to `old_period`
(17) **Else**
(18) Find the new time randomly for course j section a, `new_period`
(19) **If** `new_period` is feasible
(20) Update the current solution, `current`
(21) **Else**
(22) Set the current solution, `current` = `temp`
(23) **Break**
(24) **If** teacher replacement and course scheduling does exist
(25) Calculate the change of the objective function value Δ
(26) **If** $\Delta \leq 0$
(27) Accept the new allocation
(28) **Else**
(29) Choose a random number r_2 uniformly from [0, 1]
(30) If $r_2 < e^{-\Delta/T}$ then accept the new allocation
(31) **If** new allocation is accepted
(32) Set `temp` = `current`
(33) **If** `current` is better than best
(34) Set `best` = `current`
(35) **Else**
(36) Set the current solution, `current` = `temp`
(37) **Else**
(38) Set the current solution, `current` = `temp`

Fig. 4 Pseudocode of the improvement phase

The first operation is started by choosing course j randomly, followed by finding another teacher without violating the maximum load constraint. Two possible neighborhood structures are considered: the new teacher will be added to the list of teachers who teach course j or the new teacher will replace the teacher who has been allocated to course j. Both alternatives are selected randomly.

The second operation is to schedule the changes in teacher assignment. The new teacher is allocated to the previous day

and time periods scheduled for the previous teacher without violating the constraints. Otherwise, a new set of day and time periods has to be found. The time complexity of exploring the neighborhood of both operations is $O(|I||J||K||L||M|)$.

VI. Computational Experiments

The proposed algorithm was coded in C++ and tested on an Intel Pentium IV 2.6 GHz CPU with 512 MB RAM under the Microsoft Windows XP Operating System. Experiments were performed using the data sets from [14]. The characteristics of the data sets are summarized in Table I. Each data set consists of five randomly generated data instances.

As mentioned earlier, the initial solution is generated in the construction phase by using an integer programming approach and a simple greedy heuristic. In the improvement phase, experiments to verify the best parameter configuration for the proposed algorithm were performed. The parameters of the proposed algorithms are chosen to ensure a compromise between the computational time and the solution quality. The values of the parameters used in the computational study are summarized as follows: initial temperature (T) = 10,000, the cooling factor (α) = 0.95, the number of neighbor moves = $|I||L|$, the number of iterations = $|I||L||M|$, and $LIMIT$ = $0.05|I||L||M|$.

The software used to solve the problems is ILOG OPL Studio 4.2. Table II summarizes the average objective function values obtained and the average CPU times required (in seconds). The results from the hybrid algorithm are also presented and compared against the solutions found by OPL Studio as well.

However, the optimal solution for data instances in the 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. The OPL Studio can only provide the best known solutions for some instances and these solutions might not be optimal. For such cases, we report the best known solutions obtained for comparison purposes.

In general, the performance of the hybrid algorithm has been encouraging. In terms of CPU time, it takes less time than the OPL Studio. It is noticed that the proposed algorithm is more effective than the integer programming approach in dealing with large problems as some of the problems that could not be solved by the integer programming solver of OPL Studio can be easily solved by the proposed hybrid algorithm.

The percentage deviation of the objective value obtained by the proposed algorithm from the best known/optimal objective function value, which is defined by *Pct* = (*best known/optimal objective function value – objective function value obtained by hybrid algorithm*) / (*best known/optimal objective function value*) × 100, is calculated. We observe that the proposed heuristic is able to yield good solutions with the percentage deviation from the best known/optimal solutions being less than 18% (Table II).

TABLE I
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

TABLE II
Comparison of Best Known/Optimal Solutions and the Hybrid Algorithm's Solutions

Data set	Solution obtained by OPL Studio		Solution by the Hybrid algorithm		Average *Pct* of objective function value
	Average objective function value	Average CPU time (in seconds)	Average objective function value	Average CPU time (in seconds)	
10×20_1	7,766	1,183.11	6,612	0.42	14.87
10×20_2	7,934	1,273.15	6,536	0.41	17.48
20×30_1	10,875	9,746.36	9,554	7.14	12.46
20×30_2	13,468	36,019.99	11,362	11.00	15.55
20×40_1	13,742	86,400.00	11,884	10.31	13.55
20×40_2	16,865	86,400.00	13,932	20.36	17.70
30×60_1	20,900	86,400.00	18,610	78.00	10.17
30×60_2	24,560	86,400.00	21,008	120.34	14.96

VII. Conclusion

In this paper, we have addressed teacher assignment and course scheduling problems simultaneously and proposed a collaborative hybrid algorithm for solving this problem. The hybrid algorithm combines an integer programming approach, a greedy heuristic and a modified simulated annealing algorithm sequentially.

The experiments conducted in this paper suggest that the proposed hybrid algorithm is capable of overcoming the limitations of an integer programming approach on large data sets. The results obtained are also compared against the best known/optimal solutions generated by commercial software. We conclude that the hybrid algorithm yields good solutions within reasonable amount of computation time.

There are some possible areas of further research arising from our work. We can look into ways of improving the proposed hybrid algorithms. This would include using other types of hybrid algorithms and developing other neighborhood structures to solve the model more efficiently and yield better solutions. Finally, the mathematical programming model can always be extended to adapt to different characteristics and requirements of other universities.

References

[1] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," *SIAM Journal of Computing*, vol.5, pp. 691-703, 1976.

[2] M.W. Carter, and G. Laporte, "Recent developments in practical course timetabling," (1998), in Practice and Theory of Automated Timetabling II, Lecture Notes in Computer Science, vol.1408, ed by E. Burke, and M. Carter, New York: Springer-Verlag, pp. 3-19, 1998.

[3] R.A. Valdes, E. Crespo, and J.M. Tamarit, "Design and implementation of a course scheduling system using Tabu Search," *European Journal of Operational Research*, vol.137, pp. 512-523, 2002.

[4] S. Daskalaki, T. Birbas, and E. Housos, "An integer programming formulation for a case study in university timetabling," *European Journal of Operational Research*, vol.153, pp. 117–135, 2004.

[5] D. Abramson, M. Krishnamoorthy, and H. Dang, "Simulated annealing cooling schedules for the school timetabling problem," *Asia-Pacific Journal of Operational Research*, vol.16, pp. 1-22, 1999.

[6] J. Puchinger, and G.R. Raidl, "Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification," in IWINAC 2005, Lecture Notes in Computer Science, vol.3562, ed by J. Mira and J.R. Alvarez, New York: Springer-Verlag, pp. 41-53, 2005.

[7] R. Weare, E. Burke, and D. Elliman, "A hybrid genetic algorithm for highly constrained timetabling problems," University of Notthingham, Computer Science Technical Report No. NOTTCS-TR-1995–8, 1995.

[8] S. Petrovic, and Y. Yang, "Case-based initialisation of metaheuristics for examination timetabling," in Multidisciplinary Scheduling, Theory and Applications, ed by G. Kendall, E. Burke, S. Petrovic, and M. Gendreau, New York: Springer-Verlag, pp. 289-308, 2005.

[9] M. Chiarandini, and T. Stützle, "A landscape analysis for a hybrid approximate algorithm on a timetabling problem," TU Darmstadt, Technical Report AIDA–03–05, 2003.

[10] T.A. Duong, and K.H. Lam, "Combining constraint programming and simulated annealing on university exam timetabling," in Proceedings of International Conference RIVF'04, Hanoi, February 2004, pp. 205–210.

[11] L.T.G Merlot, N. Boland, B.D. Hughes, and P.J. Stuckey, "A hybrid algorithm for the examination timetabling problem," in Practice and Theory of Automated Timetabling IV, Lecture Notes in Computer Science, vol.2740, ed by E. Burke, and M. Carter, New York: Springer-Verlag pp. 207-231, 2002.

[12] M. Chiarandini, M. Birattari, K. Socha, and O. Rossi-Doria, "An effective hybrid algorithm for university course timetabling," *Journal of Scheduling*, vol.9, no.5,pp. 403–432, 2006.

[13] A. Gunawan, K.M. Ng, and K.L. Poh, "A mathematical programming model for a timetabling problem," in Proceedings of the International Conference on Scientific Computing, Nevada, 2006.

[14] A. Gunawan, K.M. Ng, and K.L. Poh, "An improvement heuristic for the timetabling problem (Accepted for publication)," *International Journal of Computational Science*, vol 1,no 2, pp. 162-178, 2007.

[15] S. Kirkpatrick, C.D. Gellatt, and M.P. Vecchi, "Optimization by simulated annealing," *Science*, vol 220, pp. 671-680, 1983.

Aldy Gunawan is currently a Ph.D student of the Department of Industrial and Systems Engineering, National University of Singapore. He obtained his M. Sc and M.Eng in Industrial and Systems Engineering from National University of Singapore. His research interests are in the areas of heuristic and metaheuristic algorithms, timetabling problem and other combinatorial optimization problems.

Kien Ming Ng is an assistant professor in the Department of Industrial and Systems Engineering, National University of Singapore. He obtained his PhD in Management Science and Engineering from Stanford University. His research interests are in optimization and numerical algorithms, as well as operations research applications in logistics.

Kim Leng Poh is currently an associate professor with the Department of Industrial & Systems Engineering, and the Deputy Director of Temasek Defense Systems Institute at the National University of Singapore. He received his PhD in Engineering-Economic Systems from Stanford University. His current teaching, research and consulting interests include decision analysis, investments & risk analysis, automated decision making under uncertainty & resource constraints, and large-scale optimization.