

# A MATHEMATICAL PROGRAMMING MODEL FOR A TIMETABLING PROBLEM

Aldy GUNAWAN, Kien Ming NG, Kim Leng POH  
Department of Industrial & Systems Engineering  
Faculty of Engineering  
National University of Singapore  
1 Engineering Drive 2, S(117576), Singapore  
Email: isenkm@nus.edu.sg

*Abstract* – This paper considers a timetabling problem and describes a mathematical programming model for solving it. The proposed model combines both teacher assignment and course scheduling problems simultaneously, which causes the entire model to be more complex. However, we are able to solve the model for several randomly generated data sets of sizes comparable to that encountered in an institution. The computational results for solving the model are reported together with some comparison and analysis of the optimal solutions obtained.

*Keywords*: Timetabling, integer program, mathematical programming

## 1. Introduction

Timetabling is the allocation, subject to constraints, of given resources to objects being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives [1]. One of the key applications of timetabling is in educational timetabling. Figure 1 shows the classification of educational timetabling problems.

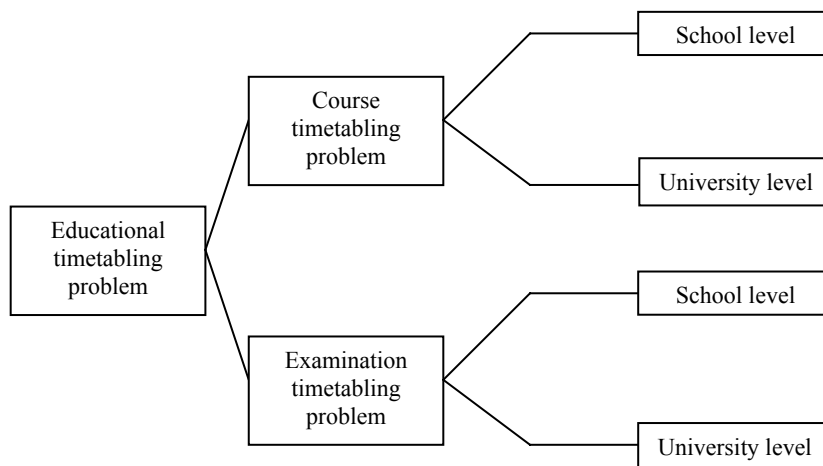


Figure 1: Classification of educational timetabling problems

Carter and Laporte [2] further classified the course timetabling problem into five subproblems: teacher assignment, class-teacher timetabling, course scheduling, student scheduling, and classroom assignment. The teacher assignment problem only focuses on allocating teachers to courses, without considering how the courses can be allocated to time periods. For the course scheduling problem, it is often assumed that the allocation of teachers to courses has been done earlier before the actual scheduling of time periods to courses.

In this paper, we propose a mathematical programming model that combines teacher assignment and course scheduling problems simultaneously at the university level. Such a timetabling problem that we address has arisen in the context of a university in Indonesia. However, the resulting model becomes more complex than if we consider teaching assignment and course scheduling separately.

## 2. The Timetabling Problem

The teacher assignment problem has been studied for many years, with one of the earliest papers being written by Andrew and Collins [3] 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. Most of the early research work did not include consideration of conflicting goals in the assignment problem. Basically, the assignment problems are formulated as linear or integer programs. The natural conflict between competing individual teachers in course-teacher assignment can be represented as a goal programming model [4]. However, such models usually do not deal with the course scheduling problem. Badri [5] proposed a model to overcome this drawback. Through 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.

The course scheduling problem has also been widely studied and many researchers have developed algorithms for solving it. Yu and Sung [6] used a genetic algorithm for solving course scheduling problems, while simulated annealing algorithms were applied to course scheduling by Abramson [7] and Abramson *et al.* [8]. Hertz [9], [10] proposed using tabu search for solving large-scale timetabling problems.

One of the most recent applications of integer programming to timetabling problems was presented by Daskalaki *et al.* [11]. We also adopt a similar approach by formulating an integer programming model that considers both teacher assignment and course scheduling simultaneously.

## 3. The Mathematical Programming Model

We define the following sets to be used in our proposed integer programming formulation:

- $I$  set of all teachers
- $J$  set of all courses
- $K$  set of all time periods
- $J_i$  set of courses that could be taught by teacher  $i$  ( $i \in I$ )
- $L_i$  maximum load (in terms of the number of courses) of teacher  $i$  ( $i \in I$ )
- $C_k$  number of classrooms available during time period  $k$  ( $k \in K$ )
- $PC_{ij}$  value given by teacher  $i$  on the preference to be assigned to teach course  $j$  ( $j \in J_i, i \in I$ )
- $PT_{ik}$  value given by teacher  $i$  on the preference to be assigned to teach in time period  $k$  ( $i \in I, k \in K$ )

We also define the following decision variables:

$$X_{ijk} = \begin{cases} 1 & \text{if teacher } i \text{ teaches course } j \text{ at time period } k \\ 0 & \text{otherwise} \end{cases} \quad (i \in I, j \in J, k \in K)$$

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

$$\text{Maximize} \quad \sum_i \sum_j \sum_k (PC_{ij} + PT_{ik}) \times X_{ijk} \quad (1)$$

subject to

$$1 \leq \sum_j \sum_k X_{ijk} \leq L_i \quad i \in I \quad (2)$$

$$\sum_i \sum_j X_{ijk} \leq C_k \quad k \in K \quad (3)$$

$$\sum_j X_{ijk} \leq 1 \quad i \in I, k \in K \quad (4)$$

$$\sum_i \sum_k X_{ijk} = 1 \quad j \in J \quad (5)$$

$$X_{ijk} = 0 \quad j \notin J, i \in I, k \in K \quad (6)$$

$$X_{ijk} \in \{0, 1\} \quad i \in I, j \in J, k \in K \quad (7)$$

The objective function (1) reflects a preference function that needs to be maximized. It refers to the preferences of assigning course  $j$  to teacher  $i$  at time period  $k$ . Thus, the course preference can not only be chosen by the teacher, but the time preference can be reflected as well. In our model, we have assumed that both preferences are equally important. However, the values for these preferences can be scaled, or the preference function can be modified according to the actual timetabling scenarios encountered.

Equation (2) ensures that each teacher has to teach at least one course, while not allowing the teacher to teach more than the maximum load of courses allowed for each teacher. Here, the load refers to the number of courses that are assigned to the teacher. Equation (3) represents the constraint on the number of classrooms available during each time period. Equation (4) ensures that each teacher can only teach at most one course at any time period. Equation (5) states that each course has to be allocated with a teacher at a particular time period. Equation (6) ensures that teachers will not be assigned courses that they are unable to teach. Finally, constraints (7) impose the 0-1 restrictions on the decision variables  $X_{ijk}$ .

## 4. Computational Results

The mathematical programming model described above was implemented using ILOG OPL Studio 3.7 on a 2.59GHz Pentium IV PC with 512MB RAM that runs in Microsoft Windows XP operating system. We randomly generated several data sets by writing a Visual C++ 6.0 program in such a way that the data sets correspond to differing values of four parameters. Here, the four parameters that were varied are the number of teachers, the number of courses, the maximum load and the number of classrooms available. These data sets have sizes that are comparable to a timetabling problem arising in a university of Indonesia, and each data set contains 5 randomly generated data instances. 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 in which each time period has a length of 3 hours.

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

$$\text{maximum load} \geq \left\lceil \frac{\text{number of courses}}{\text{number of teachers}} \right\rceil \quad (8)$$

$$\text{number of classrooms available} \geq \left\lceil \frac{\text{number of courses}}{\text{number of time periods}} \right\rceil \quad (9)$$

Table 1 shows the computational results when the number of teachers is set to 50. It summarizes the average optimal objective function values obtained and the average CPU times required to obtain the solutions.

Table 1: Computational results when number of teachers is 50

Data set	Number of teachers	Number of courses	Maximum load	Number of classrooms available	Average objective function value	Average CPU time (seconds)
50_1	50	200	4	10	34,167.40	19.30
50_2	50	200	5	10	35,645.80	16.77
50_3	50	200	6	10	35,847.60	16.93
50_4	50	200	4	15	34,797.40	15.77
50_5	50	200	4	20	34,968.40	15.40
50_6	50	250	5	13	41,185.20	29.92
50_7	50	300	6	15	48,083.00	54.81
50_8	50	350	7	18	54,139.80	119.48
50_9	50	400	8	20	59,709.80	204.99

Data set 50\_1 is used as the 50-teacher base data set by setting equations (8) and (9) to equality. For data sets 50\_2 to 50\_5, we change the value of either the maximum load or the number of classrooms available in the 50-teacher base data set. In general, we observe that some of the average CPU times required decrease when the maximum load or the number of classrooms available increases. This could be because when we increase the maximum load or the number of classrooms available, it becomes easier for equations (2) and (3) to be satisfied, which may result in increased efficiency when finding the optimal solution for these problem instances. When the maximum load or the number of classrooms available is increased, the average objective value of the optimal solution is also increased. This is because the chances for each teacher to be assigned the courses and time periods preferred will be higher.

Data sets 50\_6 to 50\_9 consider the case when the number of courses offered is increased. In order to ensure feasibility, we also increase the maximum load and the number of classrooms available by using equations (8) and (9). We notice that when the number of courses is increased, the average CPU time required increases significantly (see Figure 2).

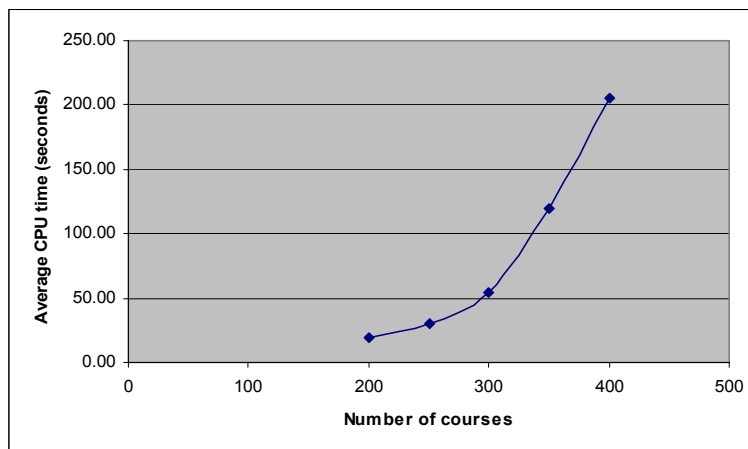


Figure 2: Plot of average CPU time for data sets 50\_1, 50\_6 to 50\_9

The next set of results looks into the effect of increasing the number of teachers to 100. We also have to adjust the maximum load and the number of classrooms available accordingly. Table 2 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 the data sets 50\_2 to 50\_9.

Table 2: Computational results when number of teachers is 100

Data set	Number of teachers	Number of courses	Maximum load	Number of classrooms available	Average objective function value	Average CPU time (seconds)
100_1	100	200	2	10	36,887.80	190.23
100_2	100	200	3	10	37,938.20	208.10
100_3	100	200	4	10	38,342.40	216.50
100_4	100	200	2	15	37,371.00	176.10
100_5	100	200	2	20	37,290.00	139.18
100_6	100	250	3	13	45,667.60	369.13
100_7	100	300	3	15	52,015.40	547.42
100_8	100	350	4	18	60,360.20	1355.55
100_9	100	400	4	20	65,259.80	1818.39

From Table 2, 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). In addition, we also 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 1 and 2.

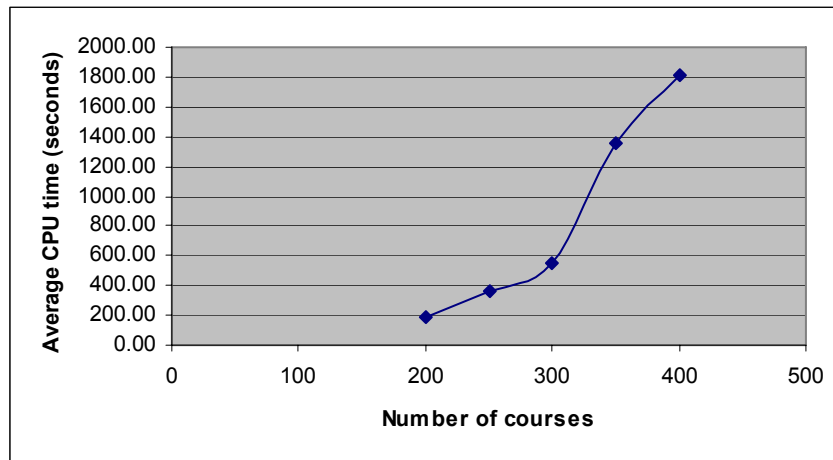


Figure 3: Plot of average CPU time for data sets 100\_1, 100\_6 to 100\_9

Another observation of interest is the load distribution of teachers in the optimal solutions with respect to the maximum load that we specify (see Table 3). The load variances obtained are reasonably small, with only a small percentage of teachers having very small load or very large load. Thus, the optimal solutions obtained also appear to have the property of sharing the load among teachers as equitably as possible.

Table 3: Load distribution of teachers

Data set	Maximum load	Average percentage of teachers having following load						Load variance
		1	2	3	4	5	6	
50_1	4	0%	0%	0%	100%	0%	0%	0.0
50_2	5	0%	2.4%	20.4%	52.0%	25.2%	0%	0.552
50_3	6	0%	1.2%	26.0%	47.6%	22.0%	3.2%	0.656
100_1	2	0%	100.0%	0%	0%	0%	0%	0.0
100_2	3	18.4%	62.8%	18.8%	0%	0%	0%	0.372
100_3	4	20%	60.6%	18.6%	0.8%	0%	0%	0.418

## 5. Conclusions

We have proposed a mathematical programming model for a timetabling problem that combines both teacher assignment and course scheduling simultaneously. We reported the computational results of solving the model for some randomly generated data sets. We also observed how the average computation time changes when a problem parameter, such as the number of teachers or courses offered, is increased.

The computational results are promising as they indicate that timetabling problems with data sizes comparable to that of an institution can be solved with the help of the model. As a future research direction, we can look into methods that would solve the model more efficiently, especially when the problem size becomes very large. Also, we can further improve the model by incorporating additional constraints that may be unique to timetabling problems in certain institutions.

## References

- [1] A. Wren. Scheduling, timetabling and rostering – A special relationship? In E. Burke and P. Ross (eds.), *Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science, Springer-Verlag New York, 1153, pp. 46-75, 1996.
- [2] M.W. Carter and G. Laporte. Recent developments in practical course timetabling. In E. Burke and M. Carter (eds.), *Practice and Theory of Automated Timetabling II*, Lecture Notes in Computer Science, Springer-Verlag New York, 1408, pp. 3-19, 1998.
- [3] G.M. Andrew and R. Collins. Matching faculty to courses. *College and University*, 46(2), pp. 83-89, 1971.
- [4] M.J. Schniederjans and G.C. Kim. A goal programming model to optimize departmental preference in course assignments. *Computers & Operations Research*, 14(2), pp. 87-96, 1987.
- [5] M.A. Badri. A two-stage multiobjective scheduling model for [faculty-course-time] assignments. *European Journal of Operational Research*, 94(1), pp. 16-28, 1996.
- [6] E. Yu and K.-S. Sung. A genetic algorithm for a university weekly courses timetabling problem. *International Transactions in Operational Research*, 9(6), pp. 703-717, 2002.
- [7] D. Abramson. Constructing school timetables using simulated annealing: sequential and parallel algorithms. *Management Science*, 37(1), pp. 98-113, 1991.
- [8] D. Abramson, M. Krishnamoorthy and H. Dang. Simulated annealing cooling schedules for the school timetabling problem. *Asia-Pacific Journal of Operational Research*, 16(1), 1-22, 1999.
- [9] A. Hertz. Finding a feasible course schedule using tabu search. *Discrete Applied Mathematics*, 35(3), pp. 255-270, 1992.
- [10] A. Hertz. Tabu search for large scale timetabling problems. *European Journal of Operational Research*, 54(1), pp. 39-47, 1991.
- [11] S. Daskalaki, T. Birbas and E. Housos. An integer programming formulation for a case study in university timetabling. *European Journal of Operational Research*, 153(1), pp. 117–135, 2004.