Operations Management Research (2019) 12:173–181
https://doi.org/10.1007/s12063-019-00146-8

A novel two-stage optimization scheme for solving university class scheduling problem using binary integer linear programming

Jilan Samiuddin[1] · Mohammad Aminul Haq[1]

Received: 23 January 2019 / Revised: 15 July 2019 / Accepted: 20 September 2019 / Published online: 20 November 2019
© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract
University Class Scheduling Problem (UCSP) is an inevitable task every university must go through prior to the commencement of a semester. The problem studied in this paper is decomposed into two stages - lab scheduling and theoretical class scheduling. This novel technique of decomposition of the problem not only allowed maximum number of free variables to stricter constraints found in lab scheduling, but also reduced the overall computational cost and combinatorial complexity. The mathematical model for the problem is formulated via Binary Integer Linear Programming (BILP) structure using data collected from the department considered in the study. Part of the contribution of this work also includes developing ways to recognize only the true variables while formulating the model. The model is then optimized using the simplex method with the objectives to optimize classroom utilization and faculty preferences while fulfilling additional constraints. Furthermore, the proposed method is compared to the traditional technique in which the model is optimized in a single stage with both true variables recognized and not recognized.

Keywords University class scheduling problem · Binary integer linear programming · Timetabling · Simplex method · Resource allocation

1 Introduction

University Class Scheduling Problem (UCSP) is a key responsibility that must be obliged to ensure a functional university. To manually schedule class routine is not only painstaking, but also prone to combinatorial error since the number of parameters and constraints involved are usually significantly large. Furthermore, manual labor cannot guarantee optimal utilization of the resources while maintaining strict constraints and achieve a feasible solution. The UCSP is class of optimization programs that is used to help universities schedule on-ground classes to lecture rooms with the objective of, for example, optimizing room utilization, subject to several constraints such as capability of a lecturer to offer a given course, and the need for specialized rooms such as computer or scientific laboratories. Solution of the realistic UCSP is necessary to ensure that the university operates efficiently allowing students to progress through their studies expeditiously by being able to register for a full load of courses for which they have met the prerequisites.

The constraints involved in the UCSP can be classified into two types:

- Hard constraints which must be satisfied under any provided circumstances. For example, a faculty cannot be present in two classrooms at the same time slot, and hence, this restriction must be obeyed as a hard constraint during the optimization process. These hard constraints form the real constraints of the problem formulation. Solutions to USCPs that do not violate the hard constraints are considered to be feasible solutions.
- Soft constraints are not as rigid like the hard constraints and should be satisfied as much as possible. For example, universities prioritize a faculty's preferred schedule, however, in order to satisfy this preference, the hard constraints cannot be breached. Therefore, the soft constraints form the objective function of the problem formulation. The computational complexity and the associated storage requirements involved in obtaining solutions that do not violate the soft constraints of a USCP often requires some relaxation of the soft constraints to obtain feasible solutions.

✉ Jilan Samiuddin
jilan@uttarauniversity.edu.bd

[1] Uttara University, House-4 & 6, Road-15, Sector-6, Uttara Model Town, Uttara, Dhaka 1230, Bangladesh

Although the scheme and general principles behind class scheduling is uncomplicated, in practical cases, the size of data is usually of substantial amount to generate significant amount of combinatorial complexity. Hence, the test data is often (over)simplified to make the solving process easy, while a practical instance of identical size may be anything but easy to solve (Murray et al. 2006; De Causmaecker & Berghe 2012).

In case of small dimensional instances, direct approaches can be used to find a solution, however, for most practical problems, finding a feasible solution using direct approaches is impossible. To tackle such practical problems, one scheme is to divide the problem into smaller segments which can be solved separately (Murray et al. 2006; Zhai et al. 2010).

Sandström (2012) divided the class scheduling problem into stages because the problem is too big to be optimized at once and computer runs out of memory. Another advantage of breaking into stages is that it becomes easier to manipulate the outcome of a step-wise optimization. However, breaking the problem into several steps does not ensure a solution as good as the solving problem at once would (Murray et al. 2006), but for scheduling problems, an optimal solution is mostly not a requirement – rather a feasible solution suffices (Sandström 2012). In this work, the optimization problem is split into two steps in order to have the option to manipulate the result of the optimization from the first step and to forward the result to the second step. Also, because of splitting the problem, memory usage is reduced significantly that minimize the possibility of the computer running out of memory.

Nanda et al. (2012) proposed an algorithm which automatically generates schedules for school lectures using a heuristic approach. The proposed algorithm is an adaptive one with the key objective to avoid clashes of lectures and subjects involving teachers only. Burke et al. (2004) used Graph-Based Hyper Heuristic (GHH) for solving timetabling problems. They investigated their approach by applying it to exam time scheduling problems (Carter et al. 1996).

Genetic algorithm (GA), being an adaptive search algorithm (Juang et al. 2007), is another popular choice in solving time scheduling problems. Abdelhalim and El Khayat (2016) presented a GA which uses a simple weighted sum formula to favor preferences of professors while handling conflicts and maximizing resource utilization. While most of the literature considered preferences of professors as an integral of the objective function (in other words as soft constraints), Abdelhalim and El Khayat considered it as an integral of the hard constraints (to a certain priority measure). Badoni et al. (2014) applied a hybrid algorithm (which comprises of GA and iterated local search to avoid being confined in local optima) to generate university course timetables for a number of class-scheduling problems. El-Sherbiny et al. (2015) merged hill climbing optimization and genetic algorithm to build a university course timetable for class-scheduling problems, and their objective was to minimize violation of any of the soft constraints.

Yazdani et al. (2017) developed a mathematical model using linear integer programming to solve UCSP, and used three different metaheuristics based on artificial immune, genetic and simulated annealing algorithms to solve the model. The objective was to maximize the instructors' preferences and minimize the number of classrooms used. Landa Silva (2003) formulated the space allocation problem as a multi-objective problem and used four metaheuristics (iterative improvement, simulated annealing, tabu search and genetic algorithms) to tackle the problem. Hybrid approaches are also adapted to tackle the multi-objective problem. Hertz (1991) also uses tabu search with the objective of reducing the number of conflicts due to courses taking place simultaneously but involving common students or teachers, or requiring the same classroom. Mooney et al. (1996) formulated class scheduling problem at Purdue University as multiple choice quadratic vertex packing (MCQVP) problem which they then solved using an enhanced local search algorithm known as the dynamic biased sampling with strategic oscillation (DBSO) algorithm. Their objective is to satisfy both instructors' and students' preferences while avoiding all sort of conflicts.

Two ant systems, the ant colony system (ACS) and the MAX-MIN ant system, are presented by Socha et al. (2003) in order to minimize the soft constraints without violating hard constraints. The soft constraints include number of classes students can take in the last time-slots of a day and probability of a student having more than one class each day. Chen and Shih (2013) applied the inertia weight and constriction versions of Particle Swarm Optimization (PSO) for UCSP, and their objective was to respect teachers' and classes' preferences, however, no hard or soft constraint is part of the formulation that respects room capacities. This aspect of prioritizing teachers is integrated in our work.

Integer programming is frequently used for scheduling problems. Ülker and Landa-Silva (2010) formulated office space allocation problem using binary integer programming. The problem was solved using CPLEX with the objective of optimizing space utilization while satisfying the hard constraints. Wasfy and Aloul (2007) used linear integer programming as well to solve UCSP using CPLEX with the aim to optimize space utilization without violating hard constraints. Thongsanit (2014) used Excel's Premium Solver to solve the mathematical model for class scheduling formulated using integer linear programming. The objective was to minimize the cost of scheduling. Aizam and Uvaraja (2015) proposed a generic model for timetabling using binary integer linear programming. They used CPLEX as solver to solve the mathematical model. However, they used medium sized random data, which they claim to be similar to realistic data, to verify their generic model. Wormald and Guimond (2012) also formulated a binary integer linear programming problem to solve for course scheduling with the objective of maximizing room utilization and "happiness coefficients" which are weights to

give a penalty or reward for certain course-room assignments. Additionally, they also created an automated system to add a new course to a given schedule.

Optimal resource allocation, time scheduling problems and alike are generally solved using linear programming since its invention. To solve linear programming (LP), several algorithms have been developed – although more recent codes use Karmarkar's Interior Point Algorithm, most computer codes for optimization still use the Simplex Method as a base. Comparing the runtime between the simplex method and the Karmarkar's method, Szabó and Kovács (2003) observed that neither is faster than the other in all problems. Moreover, they noted that the practical efficiency of both the algorithms strongly relies on the problem formulation. The simplex method has been arbitrarily adapted for this paper.

The simplex method was invented by G.B.Dantzig in the year 1963, even though his earliest published works regarding the subject dates back to 1951 (Dantzig 1951a, b). His monograph on linear programming is still an important reference since its publication (Dantzig 1963). However, many problems require integer solution sets. In this regard, Mixed Integer Linear Programming (MILP), Binary Integer Linear Programming (BILP) algorithms have been developed based on LP. For our work, the UCSP is formulated by splitting it into two binary integer linear programming problems and then solving them in two individual stages. This is done by optimizing the lab schedule in the first stage and carrying its result forward in the second stage of theory courses scheduling. While our objective of this decomposition is two-fold – to reduce the combinatorial complexity and the computational load and to address the issue of stricter constraints involved in lab scheduling – previous literature, like Sandström (2012), decomposed UCSP problems into stages only to reduce load on computer memory. Furthermore, in our work – unlike previous literature where all variables are generated and forwarded to the optimization algorithm – only true required variables are generated from the primal variables which further reduces the combinatorial complexity and the computational load. Only the true required variables are used to formulate the problem. This exploitation of variables is not found in existing literature to the best of our knowledge.

2 Problem decomposition

A university academic curriculum consists of courses which may be offered essentially within a particular department only, and hence the larger UCSP can be decomposed into a series of smaller departmental problems. This decomposition process allows retaining a sense of ownership in the timetable by individual departments, and therefore plays an important role in their approval of the automated generated schedule as the internal departmental body has all inclusive knowledge of the demands of the courses offered - the potential of an instructor to conduct a particular course, the need for specialized rooms for particular courses (the case of lab room allocations), etc. (Murray et al. 2006).

This paper introduces another decomposition step after the problem is decomposed as mentioned above; however, this step applies when there is a need of lab classes scheduling for a department and alike. In general, lab courses have stricter constraints since these require specialized lab facilities unlike theoretical courses which can be assigned to any classroom. Taking this into consideration, the problem is decomposed into two stages - lab courses and theoretical courses are optimized in the first and second stages respectively. One of the main reasons to optimize the lab courses first is because it is desirable to allocate maximum number of free variables to the stricter constraints, so that it would be more convenient for any linear optimization technique to satisfy all the stricter constraints.

3 Problem formulation

The objective of UCSP in this paper is to obtain a class schedule in order to ensure efficient allocation of resources and meeting preferences of instructors and student groups, while not violating constraints set by the university. The problem formulation using Binary Integer Linear Programming (BILP) is described in this section. Consider a department with a set of d instructors, a set of e student groups and a set of f courses for individual student groups. The courses are to be scheduled in b number of days and each day has c number of time slots. There are a numbers of rooms available for allocating the courses. The following notations are also used in the mathematical modeling:

i Index for rooms $\{1, 2, 3,...,a\}$
j Index for days $\{1, 2, 3,...,b\}$
k Index for time slots $\{1, 2, 3,...,c\}$
l Index for instructors $\{1, 2, 3,...,d\}$
m Index for student groups $\{1, 2, 3,...,e\}$
n Index for lab/theory courses of individual student groups $\{1, 2, 3,...,f\}$

Therefore, the following expression can be defined:

$$x_{ijklmn} = \begin{cases} 1 & \text{if instructor } l \text{ conducts } n\text{th course for } m\text{th student group in room } i \text{ on day } j \text{ and time slot } k, \\ 0 & \textit{otherwise} \end{cases} \quad (3.1)$$

First of all, the possible set of variables for each stage of optimization must be generated before defining the constraints. The set of variables for the first stage of optimization, V_l, i.e.

for lab scheduling, are generated by the following pseudo code:

$$\text{count} = (0)$$
$$\text{for } p = 1 \text{ to length}(L)$$
$$\text{for } q = 1 \text{ to length}(T)$$
$$\text{count} = \text{count} + 1$$
$$V_l(\text{count}, :) = \left[L(p, 1), T\left(q, 1:2\right), L\left(p, 2:4\right).\right]$$

where, L is the set of lab rooms i which must be paired with lab course n of student group m conducted by instructor l, and T is the set of all the possible day j and slot k combinations known as the time-period set. The formation of the set V_l allows the option of not adding the constraint that specifies the pairing of a lab course to a particular lab room since the information has been integrated within V_l via L.

Similarly, the set of variables for the second stage of optimization, V_t, i.e. for theory class scheduling, are generated by the following pseudo code:

$$\text{count} = (0)$$
$$\text{for } p = 1 \text{ to length}(C_A)$$
$$\text{for } q = 1 \text{ to length}(R)$$
$$\text{count} = \text{count} + 1$$
$$V_t(\text{count}, :) = \left[R(q, 1:3), C_A\left(p, 1:3\right)\right]$$

where, C_A is the set of values of l, m and n in each row - a row represents an instructor l who has been assigned to conduct nth course of student group m, and R is the set of all the possible room i, day j and slot k combinations.

In general, the following set of hard and soft constraints are considered:

- Hard constraints:

1. Same faculty cannot be allocated in the same time-period in different rooms:

   a. For lab classes scheduling:

$$\sum_{(i,l,m,n)\in Q_l} x_{ijklmn} \le 1, \quad \forall (j,k)\in T, l \tag{3.2}$$

where, $Q_l = \{(i, l, m, n) |\, V_l(p, 4) = l \,\&\, p \in \{1, 2, \ldots, r_l\}\}$, and r_l is the number of rows in V_l.

   b. For theory classes scheduling:

$$\sum_{(i,l,m,n)\in Q_t} x_{ijklmn} \le 1, \forall (j,k)\in T, l \tag{3.3}$$

where, $Q_t = \{(i, l, m, n) |\, V_t(q, 4) = l \,\&\, q \in \{1, 2, \ldots, r_t\}\}$, and r_t is the number of rows in V_t.

2. Same student group cannot be allocated in the same time-period in different rooms:

   a. For lab classes scheduling:

$$\sum_{(i,l,m,n)\in Q_l} x_{ijklmn} \le 1, \quad \forall (j,k)\in T, m \tag{3.4}$$

where, $Q_l = \{(i, l, m, n) |\, V_l(p, 5) = m \,\&\, p \in \{1, 2, \ldots, r_l\}\}$.

   b. For theory classes scheduling:

$$\sum_{(i,l,m,n)\in Q_t} x_{ijklmn} \le 1, \forall (j,k)\in T, m \tag{3.5}$$

where, $Q_t = \{(i, l, m, n) |\, V_t(q, 5) = m \,\&\, q \in \{1, 2, \ldots, r_t\}\}$.

3. Each course must be assigned to a particular time slot in one room:

   a. For lab classes scheduling:

$$\sum_{(j,k)\in T} x_{ijklmn} = 1, \forall (i,l,m,n)\in V_{lc} \tag{3.6}$$

where, $V_{lc} = V_l(p, [1, 4:6]) \,\&\, p \in \{1, 2, \ldots, r_l\}$,

   b. For theory classes scheduling:

$$\sum_{(i,j,k)\in R} x_{ijklmn} = 1, \quad \forall (l,m,n)\in V_{tc} \tag{3.7}$$

where, $V_{tc} = V_t(q, [4:6]) \,\&\, q \in \{1, 2, \ldots, r_t\}$.

4. Each time-period for each room can fit up to one course:

   a. For lab classes scheduling:

$$\sum_{(l,m,n)\in Q_x} x_{ijklmn} \le 1, \quad \forall (j,k)\in T, i \tag{3.8}$$

where, $Q_x = \{(l, m, n) |\, V_l(p, 1) = i \,\&\, p \in \{1, 2, \ldots, r_l\}\}$.

   b. For theory classes scheduling:

$$\sum_{(l,m,n)\in Q_y} x_{ijklmn} \le 1, \quad \forall (i,j,k)\in R \tag{3.9}$$

where, $Q_y = \{(l, m, n) |\, V_t(p, 1) = i \,\&\, p \in \{1, 2, \ldots, r_t\}\}$.

5. Room capacity must exceed student groups' enrollment:

$$\sum_{(i,m)\in R} \sum_{(j,k,l,n)\in Z} x_{ijklmn} = 0, \tag{3.10}$$

where, $Z = \{(j, k, l, n) | V_t(p, (1, 5)) = (i, m) \,\&\, p \in \{1, 2, \ldots, r_t\}\}$, and R is the set of rooms and student groups which cannot be paired together. It is assumed that all lab room facilities can accommodate maximum number of students enrolled in a student group, and hence eq. (3.10) is not applicable for stage one optimization.

However, hard constraints will not be the same for both the stages. In the first stage of optimization, the following extra hard constraint is added - each lab course must be assigned to its required lab room. But this constraint does not require any mathematical formulation as the previous ones since the inclusion of L in V_l automatically takes care of this particular constraint.

- Soft constraints:

1. Student groups should be distributed efficiently among the available rooms, i.e., minimize c_{im}, where c_{im} is the ratio between capacity of room i to number of students enrolled in student group m.
2. Instructors' preferences should be considered.

$$H_{jkl} = \begin{cases} 1 & \text{if instructor } l \text{ is happy,} \\ 2 & \text{if instructor } l \text{ is okay,} \\ 3 & \text{otherwise.} \end{cases} \tag{3.11}$$

In the second stage of optimization, along with the five hard constraints mentioned above, the followings are also added, which result from the first stage of optimization:

1. Instructors assigned in a particular time period after the first stage of optimization cannot be assigned in the same time period in the second stage of optimization.
2. Student groups assigned in a particular time period after the first stage of optimization cannot be assigned in the same time period in the second stage of optimization.

The above mentioned constraints can be implemented by the following mathematical expressions:

$$V_t = V_t(:, [2, 3, 4]) \backslash S_l(:, [2, 3, 4]),$$
$$V_t = V_t(:, [2, 3, 5]) \backslash S_l(:, [2, 3, 5]),$$

where, S_l is the set of solution i, j, k, l, m and n obtained from optimizing the first stage. This can be implemented using the following pseudo code:

$$\text{index\_Vt\_ins} = \text{intersect}(V_t(:, [2, 3, 4]), S_l(:, [2, 3, 4]))$$
$$V_t = \text{delete}(V_t, \text{index\_Vt\_ins})$$
$$\text{index\_Vt\_sg} = \text{intersect}(V_t(:, [2, 3, 5]), S_l(:, [2, 3, 5]))$$
$$V_t = \text{delete}(V_t, \text{index\_Vt\_sg})$$

In addition to the above mentioned constraints, additional constraints may be applicable depending on a particular department or a university. This will be addressed further in section 6.The objective of the optimization process is as follows:

$$\text{minimize} \quad \left(c_{im} H_{jkl} x_{ijklmn}\right). \tag{3.12}$$

In summary, the problem is formulated in two steps - the objective function (3.12) (comprising of the soft constraints) being the same for both the steps. However, the hard constraints (3.2), (3.4), (3.6) and (3.8) are applicable for the first stage of optimization, whereas, the hard constraints (3.3), (3.5), (3.7), (3.9) and (3.10) are applicable for the second stage of optimization.

4 An illustrative example

In order to comprehend the problem formulation in section 4, an example is illustrated in this section. In this example, a department consisting of three instructors and three student groups, each student group having 31, 27 and 38 number of students respectively, is considered. The department has two lab rooms (can accommodate maximum number of students) and two general rooms (with capacity to accommodate 40 students and 35 students respectively) to distribute its courses - two theory courses and one lab course for each student group. These are to be allocated within 2 days with 2 slots each. The course distributions are shown in Table 1. Three preferences for each instructor are listed in Table 2, with two preferences that they are "happy" ($H_{jkl} = 1$) with, and one preference that they are "okay" ($H_{jkl} = 2$) with.

The problem is optimized in two stages. The first stage, in which the lab courses are to be distributed, has the following sets of hard constraints obtained from eqs. (3.2), (3.4), (3.6) and (3.8) respectively:

$$\left.\begin{aligned} x_{111131} + x_{211131} &\le 1 \\ x_{112131} + x_{212131} &\le 1 \\ &\vdots \\ x_{122311} &\le 1 \\ x_{121311} &\le 1 \end{aligned}\right\} \tag{4.1}$$

$$\left.\begin{aligned} x_{111311} &\le 1 \\ x_{112311} &\le 1 \\ &\vdots \\ x_{121131} + x_{221131} &\le 1 \\ x_{122131} + x_{222131} &\le 1 \end{aligned}\right\} \tag{4.2}$$

$$\left.\begin{aligned} x_{111311} + x_{112311} + x_{121311} + x_{122311} &= 1 \\ x_{211221} + x_{212221} + x_{221221} + x_{222221} &= 1 \\ x_{111131} + x_{112131} + x_{121131} + x_{122131} + & \\ x_{211131} + x_{212131} + x_{221131} + x_{222131} &= 1 \end{aligned}\right\} \tag{4.3}$$

Table 1 Course distribution

Instructor (l)	Student group (m)	Theory course (n)	Lab course (n)	Corresponding lab room (i)
1	1	1	–	–
3	1	2	–	–
1	2	1	–	–
2	2	2	–	–
2	3	1	–	–
3	3	2	–	–
3	1	–	1	1
2	2	–	1	2
1	3	–	1	1 or 2

$$\left.\begin{aligned} x_{111311} + x_{111131} &\le 1 \\ x_{112311} + x_{112131} &\le 1 \\ &\vdots \\ x_{221221} + x_{221131} &\le 1 \\ x_{222221} + x_{222131} &\le 1 \end{aligned}\right\} \tag{4.4}$$

The objective function for the first stage, obtained from eq. (3.12), is as follows:

$$\begin{aligned} \text{minimize } & (x_{111311} + 2x_{112311} + x_{121311} + 3x_{122311} \\ & + 2x_{211221} + \ldots + x_{212131} + 2x_{221131} + 3x_{222131}) \end{aligned} \tag{4.5}$$

However, because of the assumption that lab rooms have the capacity to accommodate maximum number of students allowed in a student group, $c_{im} = 1$ is considered while formulating eq. (4.5). After the completion of first stage of the optimization process, result obtained is shown in Table 3. The cells in the table show values of l, m and n.

It should be noted that even though the first expected variable is $x_{ijklmn} = x_{111111}$, it did not appear in eq. (4.5) – this variable corresponds to instructor 1 ($l = 1$) conducting lab course 1 ($n = 1$) of student group 1 ($m = 1$) which is not a possible variable since this lab course has not been allocated to instructor 1, and thus has been eliminated. Similarly, all other variables with $l = 1$, $m = 1$ and $n = 1$ are eliminated regardless of the values of i, j and k. However, since instructor 3 has been appointed to conduct lab course 1 ($n = 1$) of student group 1 ($m = 1$) all the variables (e.g. x_{111311}, x_{112311}) with $l = 3$, $m = 1$ and $n = 1$ have been retained in eq. (4.5).

Table 2 Preferences of the instructors

Instructor (l)	$H_{jkl} = 1$ (j, k)	$H_{jkl} = 2$ (j, k)
1	1,1	2,1
	1,2	–
2	2,1	1,1
	2,2	–
3	1,1	1,2
	2,1	–

Table 3 Lab schedule as a result of the first stage of optimization

Lab 1	Slot 1	Slot 2	Lab 1	Slot 1	Slot 2
Day 1	3,1,1	1,3,1	Day 1		
Day 2			Day 2		2,2,1

Now, from eqs. (3.3), (3.5), (3.7), (3.9) and (3.10), the following sets of hard constraints are obtained for the second-stage of optimization respectively:

$$\left.\begin{aligned} x_{111121} + x_{211121} &\le 1 \\ x_{121111} + x_{221111} + x_{121121} + x_{221121} &\le 1 \\ &\vdots \\ x_{121312} + x_{221312} + x_{121332} + x_{221332} &\le 1 \\ x_{122312} + x_{222312} + x_{122332} + x_{222332} &\le 1 \end{aligned}\right\} \tag{4.6}$$

$$\left.\begin{aligned} x_{112312} + x_{212312} &\le 1 \\ x_{121111} + x_{221111} + x_{121312} + x_{221312} &\le 1 \\ &\vdots \\ x_{121231} + x_{221231} + x_{121332} + x_{221332} &\le 1 \\ x_{122332} + x_{222332} &\le 1 \end{aligned}\right\} \tag{4.7}$$

$$\left.\begin{aligned} x_{121111} + x_{122111} + x_{221111} + x_{222111} &= 1 \\ x_{112312} + x_{121312} + x_{122312} + x_{212312} & \\ + x_{221312} + x_{222312} &= 1 \\ &\vdots \\ x_{111231} + x_{121231} + x_{211231} + x_{221231} &= 1 \\ x_{121332} + x_{122332} + x_{221332} + x_{222332} &= 1 \end{aligned}\right\} \tag{4.8}$$

$$\left.\begin{aligned} x_{111121} + x_{111222} + x_{111231} &\le 1 \\ x_{112312} + x_{112222} &\le 1 \\ &\vdots \\ x_{221111} + x_{221312} + x_{221121} + x_{221222} & \\ + x_{221231} + x_{221332} &\le 1 \\ x_{222111} + x_{222312} + x_{222332} &\le 1 \end{aligned}\right\} \tag{4.9}$$

$$x_{211231} + x_{221231} + x_{221332} + x_{222332} = 0 \tag{4.10}$$

The objective function for the second stage, obtained from eq. (3.12), is as follows:

$$\begin{aligned} \text{minimize } & (2.58x_{121111} + 3.87x_{122111} + 2.26x_{221111} \\ & + 3.39x_{222111} + 2.58x_{112312} + \ldots + 3.16x_{122332} \\ & + 0.92x_{221332} + 2.76x_{222332}) \end{aligned} \tag{4.11}$$

Table 4 Theory classes schedule as a result of the second stage of optimization

Room 1	Slot 1	Slot 2	Room 2	Slot 1	Slot 2
Day 1	2,3,1	3,1,2	Day 1	1,2,1	
Day 2	3,3,2		Day 2	2,2,2	1,1,1

Table 5 The proposed method in contrast to other conventional techniques

	Proposed method (combined stages)	Single-stage with true variables only	Single-stage with all variables inclusive
Number of variables	16,241	22,302	4,814,145
Optimization time (s)	6.72	43.09	System ran out of memory (cannot be solved)

As an example, the coefficient of x_{121111} (sequentially, the subscript corresponds to room 1, day 2, slot 1, instructor 1, student group 1 and course 1 of student group 1) is the product of c_{11} and h_{211} where,

$$c_{11} = \frac{\text{Capacity of room 1}}{\text{Number of students in student group 1}} = \frac{40}{31},$$

$h_{211} = 2 \ldots (\text{Table 2}).$

After the completion of the second stage of the optimization process, result obtained is shown in Table 4.

5 Case study

The department of Electrical and Electronic Engineering at Uttara University has to accommodate more than 1000 students, divided into 31 student groups. There are 51 instructors available to conduct a total of 184 courses. The university has allocated 19 general rooms and 10 engineering labs to distribute the courses. All of these courses need to be assigned in the available rooms within 9 slots distributed in 3 days. The department requires that each student group must be scheduled for 2 days including day 2, however, there are no such constraint for the instructors. This additional departmental hard constraint including the hard constraints mentioned in section 4 are considered in the case study. The additional constraints can be formulated as follows:

1. For lab classes scheduling:

$$\sum_{(j,m)\in D} \sum_{(i,k,l,n)\in Y} x_{ijklmn} = 0, \tag{5.1}$$

where, $Y = \{(i, k, l, n) | V_l(p, (2, 5)) = (j, m) \,\&\, p \in \{1, 2, \ldots, r_l\}\}$, and,

2. For theory classes scheduling:

$$\sum_{(j,m)\in D} \sum_{(i,k,l,n)\in W} x_{ijklmn} = 0, \tag{5.2}$$

Table 6 Performance parameters after optimizing "Summer 2018" class scheduling problem

% of happy slots satisfied	92.21
% of ok slots satisfied	68.42
Average room efficiency (%)	93.67

where, $W = \{(i, k, l, n) | V_t(p, (2, 5)) = (j, m) \,\&\, p \in \{1, 2, \ldots, r_t\}\}$, and D is the set of days and student groups which cannot be paired together. It should be noted that set D has not been created based on any algorithm or logic, although it has a definite impact on the cost function. However, the effect of set D on the cost function is out of scope for this paper and hence not considered as part of the study.

Using simple survey forms, the instructors were asked to fill in their preferred class schedule - namely "happy" slots and "okay" slots, and all other slots are considered to be "undesirable" slots. Data was collected from the department coordinator regarding course distribution in attempt to solve the scheduling problem for the semester of Summer 2018.

This case study, without any alteration, produces close to five million variables; not only is this computationally expensive, rather seems to be impractical as well. However, recognizing the true required variables from the primal variables set in a systematic way, as described in section 4, overwhelmingly reduces the number of variables - the number of variables are reduced to a few thousands from millions. As a result, the computational time is significantly truncated. Furthermore, the proposed method is also compared to the conventional single-stage optimization technique for class scheduling problems with only true required variables. Table 5 illustrates the improvements achieved by the proposed method in terms of memory usage (i.e. number of variables) and computational time for optimization. Clearly, the proposed method

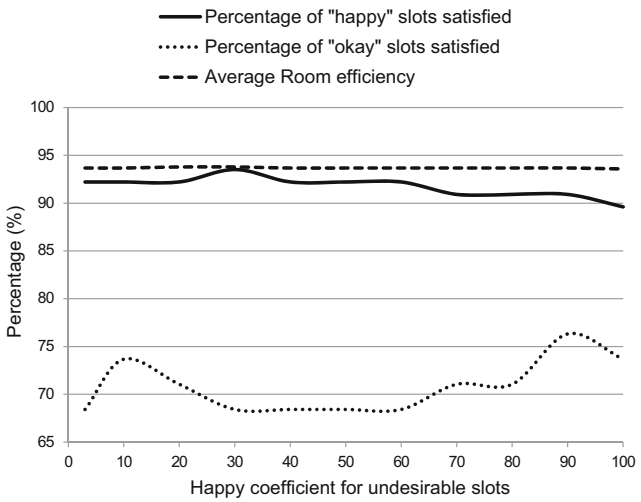

Fig. 1 Change in performance parameters with respect to change in happy coefficient for undesirable slots

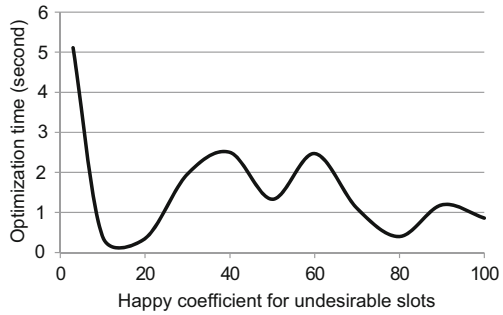

Fig. 2 Optimization time of the proposed method versus happy coefficient for undesirable slots

outperforms the conventional techniques in both the aspects. All of these were implemented in Python using a laptop computer with the following specifications: 2.40 GHz processor, 8.00 GB RAM and 64-bit operating system.

Table 6 shows the obtained performance parameters using the proposed method. However, these values, including the optimization time, are correlated to the value of happy coefficient for undesirable slots as shown in Figs. 1 and 2. In general, the performance parameters are prioritized and need to be maximized. As can be seen from Fig. 1, $H_{jkl} = 90$ for undesirable slots gives the maximum performance for the case study. It can also be noted that there is little or no change in the average room efficiency which is intuitional.

Although, when the class-scheduling problem is centralized instead of leaving it to the individual departments, the optimization time will drastically increase due to increased number of variables. In such scenario, least computational time along with maximized objective function are desirable. Thus, a new performance parameter η is proposed and analyzed:

$$\eta = \frac{w_1 \times (\%\text{of happy slots satisfied}) + w_2 \times (\%\text{of happy slots satisfied}) + w_3 \times \text{Average room efficiency}}{\text{optimization time}}.$$

where, w_1, w_2 and w_3 are weights put according to preferences. Fig. 3 shows that $H_{jkl} = 18$ for undesirable slots gives the highest value of η with $w_1 = 1$, $w_2 = 0.5$ and $w_3 = 1$.

6 Conclusion

In this paper, the UCSP was mathematically formulated using BILP and was then optimized using the simplex method. A novel scheme was developed that reduces the computational burden significantly by dividing the process into two stages. A real class scheduling problem was then solved using the proposed method to evaluate its performance. It outperformed the conventional techniques by optimizing the problem with remarkably lower computational cost. Furthermore, the proposed method exhibited satisfactory performance in maintaining faculty preferences and room efficiencies. Although the problem was solved using customized program in Python, there are several dedicated software packages devoted to ILP, MILP and BILP optimization such as LINDO and the IBM OSL (Optimization subroutine library). These are expensive commercial options – usually having higher computing power – which can be explored in order to compare computational times and limitations such as solving the UCSP problem in single-stage with all variables inclusive. However, due to the limitations of this work, the comparison could not be carried out.

Changes in the performance parameters due to change in the happy coefficient for undesirable slots is analyzed and it is observed that the value of the coefficient can alter the performance of the solution. However, this is not sufficient to comprehend a concrete idea of how to establish an optimal solution against the change in happy coefficient for undesirable slots. Since this is out of the scope of this work, for future investigations, a detailed analysis can be carried out to address the performance of UCSP with respect to happy coefficient for undesirable slots. Also, a new performance parameter η has also been introduced that requires more probing to enhance the overall performance.

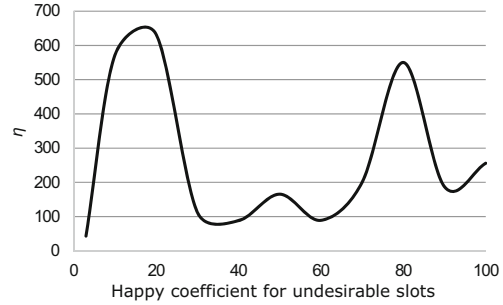

Fig. 3 η versus happy coefficient for undesirable slots

Acknowledgements We would like to thank Center for Research & Training (CRT), Uttara University, for funding this work.

References

Abdelhalim EA, El Khayat GA (2016) A utilization-based genetic algorithm for solving the university timetabling problem (uga). Alexandria Engineering Journal 55(2):1395–1409. https://doi.org/10.1016/j.aej.2016.02.017

Aizam NAH, Uvaraja V (2015) Generic model for timetabling problems by integer linear programming approach. World Academy of Science, Engineering and Technology, International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering 9(12):668–675. https://doi.org/10.5281/zenodo.1110215

Badoni RP, Gupta D, Mishra P (2014) A new hybrid algorithm for university course timetabling problem using events based on groupings of students. Comput Ind Eng 78:12–25. https://doi.org/10.1016/j.cie.2014.09.020

Burke EK, Meisels A, Petrovic S, Qu R (2004) A graph-based hyper heuristic for timetabling problems. Computer science technical report no. NOTTCS-TR-2004-9, University of Nottingham. https://doi.org/10.1016/j.ejor.2005.08.012

Carter MW, Laporte G, Lee SY (1996) Examination timetabling: algorithmic strategies and applications. J Oper Res Soc 47(3):373–383. https://doi.org/10.1057/jors.1996.37

Chen RM, Shih HF (2013) Solving university course timetabling problems using constriction particle swarm optimization with local search. Algorithms 6(2):227–244. https://doi.org/10.3390/a6020227

Dantzig GB (1951a) Application of the simplex method to a transportation problem. Activity analysis and production and allocation. Koopmans, T.C., Ed., John Wiley and Sons, New York, 359-373

Dantzig GB (1951b) A proof of the equivalence of the programming problem and the game problem. Activity analysis of production and allocation, no 13:330–338

Dantzig GB (1963) Linear programming and extensions. Princeton landmarks in mathematics and physics

De Causmaecker P, Berghe GV (2012) Towards a reference model for timetabling and rostering. Ann Oper Res 194(1):167–176. https://doi.org/10.1007/s10479-010-0721-2

El-Sherbiny MM, Zeineldin RA, El-Dhshan AM (2015) Genetic algorithm for solving course timetable problems. International Journal of Computer Applications 124(10). https://doi.org/10.5120/ijca2015905408

Hertz A (1991) Tabu search for large scale timetabling problems. Eur J Oper Res 54(1):39–47. https://doi.org/10.1016/0377-2217(91)90321-L

Juang YS, Lin SS, Kao HP (2007) An adaptive scheduling system with genetic algorithms for arranging employee training programs. Expert Syst Appl 33(3):642–651. https://doi.org/10.1016/j.eswa.2006.06.010

Landa Silva JD (2003) Metaheuristic and multiobjective approaches for space allocation. Ph.D. dissertation, University of Nottingham

Mooney EL, Rardin RL, Parmenter W (1996) Large-scale classroom scheduling. IIE Trans 28(5):369–378. https://doi.org/10.1080/07408179608966284

Murray K, Müller T, Rudová H (2006) Modeling and solution of a complex university course timetabling problem. International conference on the practice and theory of automated timetabling. Springer, pp. 189–209. https://doi.org/10.1007/978-3-540-77345-0_13

Nanda A, Pai MP, and Gole A (2012) An algorithm to automatically generate schedule for school lectures using a heuristic approach. International journal of machine learning and computing, vol. 2, no. 4, p. 492. https://doi.org/10.7763/IJMLC.2012.V2.174

Sandström V (2012) Scheduling of teaching resources and classes using mixed integer linear programming. Aalto University, School of Science

Socha K, Samples M, Manfrin M (2003) Ant algorithms for the university course timetabling problem with regard to the state-of-the-art. Workshops on applications of evolutionary computation. Springer, pp. 334–345. https://doi.org/10.1007/3-540-36605-9_31

Szabó Z, Kovács M (2003) On interior-point methods and simplex method in linear programming. An St Univ Ovidius Constanta 11(2):155–162

Thongsanit K (2014) Solving the course-classroom assignment problem for a university. Silpakorn University Science and Technology Journal 8(1):46–52. https://doi.org/10.14456/sustj.2014.3

Ülker Ö, Landa-Silva D (2010) A 0/1 integer programming model for the office space allocation problem. Electron Notes Discrete Math 36:575–582. https://doi.org/10.1016/j.endm.2010.05.073

Wasfy A and Aloul F (2007) Solving the university class scheduling problem using advanced ilp techniques. IEEE GCC Conference

Wormald R, Guimond C (2012) Creating a more efficient course schedule at wpi using linear optimization. Major Qualifying Rep, Worcester Polytechnic Institute, Worcester

Yazdani M, Naderi B, Zeinali E (2017) Algorithms for university course scheduling problems. Tehnicki Vjesnik-Technical Gazette 24:241–247. https://doi.org/10.17559/tv-20130918133247

Zhai Y, Sun S, Wang J, Guo S (2010) A heuristic algorithm for large-scale job shop scheduling based on operation decomposition using bottleneck machine. Management and service science (MASS), 2010 international conference on. IEEE, pp. 1–4. https://doi.org/10.1109/ICMSS.2010.5576038

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.