Optimization Methods and Software

ISSN: 1055-6788 (Print) 1029-4937 (Online) Journal homepage: http://www.tandfonline.com/loi/goms20

An integer program and a hybrid genetic algorithm for the university timetabling problem

Xuehao Feng, Yuna Lee & Ilkyeong Moon

To cite this article: Xuehao Feng, Yuna Lee & Ilkyeong Moon (2017) An integer program and a hybrid genetic algorithm for the university timetabling problem, Optimization Methods and Software, 32:3, 625-649, DOI: 10.1080/10556788.2016.1233970

To link to this article: http://dx.doi.org/10.1080/10556788.2016.1233970

Published online: 29 Sep 2016.

Submit your article to this journal

Article views: 65

View related articles

View Crossmark data

Optimization Methods & Software, 2017
Vol. 32, No. 3, 625–649, http://dx.doi.org/10.1080/10556788.2016.1233970

An integer program and a hybrid genetic algorithm for the university timetabling problem

Xuehao Feng[a], Yuna Lee[b] and Ilkyeong Moon[c]*

[a] *Ocean College, Zhejiang University, Hangzhou, China;* [b] *LG Electronics, Gyeonggi-do, Republic of Korea;* [c] *Department of Industrial Engineering, Seoul National Unviersity, Seoul, Republic of Korea*

(*Received 18 December 2014; accepted 3 September 2016*)

The university timetabling problem (UTP) has been studied by numerous research groups for decades. In addition to addressing hard and soft constraints, we extend the UTP by considering consecutiveness and periodicity constraints of multi-session lectures, which are common in many eastern Asian universities. Because schedulers can decide the consecutiveness and periodicity constraints for the multi-session lectures within a limited ratio, we consider these novel decision variables in our model. We develop a mixed integer linear program for the UTP. For the analysis, we convert the UTP into the three-dimensional container packing problem (3DCPP) and create a hybrid genetic algorithm (HGA), which has been shown to be efficient in solving the 3DCPP. We also develop a tabu search algorithm based on the existing UTP literature and compare the findings with that of our HGA. The results show that our HGA obtains a better solution than the tabu search algorithm in a reasonable amount of time.

Keywords: university timetabling problem; mixed integer linear program; periodicity constraint; consecutiveness constraint; hybrid genetic algorithm

1. Introduction

1.1 *Background and purpose*

The university timetabling problem (UTP) is used for optimally assigning lectures to timeslots as defined by days and periods within classrooms. The purpose is to find feasible assignment of lectures that leads to the optimal objective. Each lecture consists of a certain number of sessions in every week. Each session of any an assigned lecture occupies exactly one day-period-classroom timeslot. Hard and soft constraints are involved in the UTP. Violating hard constraints makes the solution infeasible, while the violation of soft constraints leads to penalty costs. For example, one hard constraint states that, at most one session of a lecture can be assigned to one room-period-day slot. An example of a soft constraint is the avoidance of time conflicts between two lectures that appeal to the same students. Manual solutions of real-world UTPs require many man-hours of work, and the scale of the problems and the complexity of the constraints often result in errors. The UTP should be solved repeatedly every semester with minor modifications of some constraints and with a new set of lectures. Automated solutions of UTPs can save time and also satisfy all constraints simultaneously.

*Corresponding author. Email: ikmoon@snu.ac.kr

The UTP has been a controversial issue for decades and has been widely studied by many research groups. In 2002, the First International Timetabling Competition (ITC2002) was held by the International Metaheuristic Network. In 2007, the Second International Timetabling Competition (ITC2007) was held. It consisted of three tracks: an examination, a post-enrolment, and a curriculum-based timetabling problem. The latter was first introduced at the ITC2007 and many researchers started to focus on it. The Third International Timetabling Competition was held from 2011 to 2012 and handled the high school timetabling problem.

Nowadays, many research groups solve the UTP by reflecting upon real-world constraints and considering the general features of the UTP, including fixed timeslots, limited numbers of classrooms, restricted classroom capacities, and non-overlapping lectures. This paper focuses on the UTP with realistic hard and soft constrains in most universities in China and South Korea. In our UTP, multi-session lectures can be scheduled based on either consecutiveness or periodicity, which are set as hard constraints. If a lecture is scheduled based on the consecutiveness constraint, then sessions of that lecture in one cycle (e.g. one week) should be consecutively finished in the same room. In such an arrangement, the professor and students have a compact itinerary but also an increased daily workload. If a lecture is scheduled through the periodicity constraint, then the lecture should not occupy more than one period in one day or two consecutive days. Both of these types of arrangements reflect realistic practices of time table design in Chinese and South Korean universities. Four realistic soft constraints are also considered in our UTP (see Section 2 for details).

This study contributes to the existing literature in two ways. First, we consider consecutiveness and periodicity constraints of multi-session lectures, which are common in many eastern Asian universities. Because schedulers in those universities can decide the consecutiveness and periodicity constraints for multi-session lectures within a limited ratio, we consider these novel decision variables in our model. By allowing for decision-making variability, our UTP reveal more flexibility and managerial insights. Second, we analyse the UTP by converting it to the three-dimensional container packing problem (3DCPP). We consider day, period, and room as the three dimensions of one container and consider the lectures as different sized items to be assigned into the container. With this approach, we develop an efficient hybrid genetic algorithm (HGA) based on algorithms for the 3DCPP.

This paper presents discussions and evaluations of mixed integer linear programming (MILP) and an HGA to solve a UTP. The model reflects consideration of two real-world constraints: consecutiveness and periodicity. MILP models allow for easily added and changed constraints and can yield exact solutions that satisfy all the constraints concurrently. However, because the UTP is NP-hard [16], we developed an HGA that can derive solutions applied to large problems.

1.2 *Related works*

In this section, we review and summarize existing studies on the UTP to reveal our contribution.

Since the 1960s, much research has been devoted to solving the UTP of various descriptions and through many methodologies. There is a large body of literature on different types of UTP. The well-known ITC 2007 involved a UTP with several hard and soft constraints. Socha *et al.* [46] generated 11 timetabling problem data sets, and using them as benchmarks, various research groups have generated additional heuristic and metaheuristic methods to solve the UTP. A case study of the Department of Electrical and Computer Engineering at the University of Patras is an example of research on real-world problems [13]. The UTP was based on consecutive lectures that could be split into two or three sessions as well as repetitive lectures held several times a week. Both Innet and Nuntasen [25] and Schimmelpfeng and Helber [45] conducted experiments using real-world data from Thai University and the School of Economics and Management at Hannover University, respectively. When constructing timetables, Innet and Nuntasen [25] focused on students' spare time between lectures and during a lunch break, and Schimmelpfeng and Helber [45] focused on teachers' preferences for days, periods, breaks, consecutive lectures, and teaching groups. Kalender *et al.* [27] also solved a curriculum-based course timetabling problem at Yeditepe University, which features both compulsory and optional lectures.

In early studies, the majority of researchers utilized integer programming (IP) to construct timetables, while computer scientists used linear algorithms to solve the UTP [4,33]. Recently, MirHassani [40] and Lach and Lübbecke [32] developed IPs to look at case studies of small problems. Daskalaki *et al.* [13] developed a novel 0–1 IP to solve the UTP, but for the large case of an engineering department schedule, they used simplified mathematical expressions of the constraints. In addition, they did not consider that some lectures should be assigned according to periodicity. Daskalaki and Birbas [12] developed a two-stage relaxation procedure to solve the IP efficiently for the UTP. All the methods cited could not yield optimum solutions for large, NP-hard problems.

In general, various heuristic and metaheuristic algorithms have been studied to solve large problems, which are relatively compatible with real-world situations. The research showing solutions to the UTP by heuristic and metaheuristic algorithms covers many years and various methods. Tripathy [48] conducted the Lagrangean relaxation as a large-scale 0–1 IP problem. Čangalović and Schreuder [9] solved a UTP with a graph colouring technique. Hertz [24], Costa [11] as well as Lü and Hao [37] have used tabu search algorithms to solve the UTP. Mulvey [42] made a heuristic algorithm using the network-based optimization approach. Dowsland [19], Abramson [3], and Dige *et al.* [18] developed a simulated annealing (SA) approach. Bellio *et al.* [7] proposed an SA algorithm with feature-based tuning for the curriculum-based course time tabling problem. Constraint-based reasoning was applied by Deris *et al.* [17] and Kang and White [29]. McMullan [39] introduced an extended version of the great deluge algorithm; an electromagnetism-like mechanism with the great deluge algorithm was subsequently developed by Turabieh *et al.* [49]. De Causmaecker *et al.* [14] developed a decomposed metaheuristic approach with the graphical presentation of a solution space for a real-world UTP. Zhang *et al.* [52] developed a simulated annealing algorithm with a new neighbourhood structure to solve the high school timetabling problems. Socha *et al.* [46] solved the UTP by applying ant colony algorithms. The harmony search algorithm, a new metaheuristic mimicking the improvization process of musicians, was employed by Al-Betar and Khader [5] as well as Wahid and Hussin [50]. Abdullah *et al.* [2] developed a hybrid metaheuristic that combines an electromagnetic-like mechanism and the great deluge algorithm. Kalender *et al.* [27] applied an iterative selection hyper-heuristic (a combination of SA move acceptance and learning heuristic selection methods) to solve the UTP at Yeditepe University, and Soria-Alcaraz *et al.* [47] proposed hyper-heuristics based on an iterated local search procedure that autonomously combines a set of move operators. Liu *et al.* [36] developed a clique-based algorithm for the UTP with hard constraints. For a comprehensive survey of approaches for the UTP, see Babaei *et al.* [6].

Erben and Keppler [21] were the first to utilize the genetic algorithm (GA) to tackle the UTP. Further studies on the GA approach have been conducted [1,10,20,25,34,35]. Rossi-Doria *et al.* [44] compared five different metaheuristics and concluded that the conventional GA cannot always achieve a solution better than other metaheuristics. The need for enhanced GAs has encouraged others to find better solutions. The HGA method, which combines the advantages of both heuristic and GAs, enhances solutions [51]. The HGA has been further developed into different mechanisms by many researchers. For example, Jat and Yang [26] proposed a two-phase HGA to solve the post-enrolment course timetabling problem. In the first phase, the guided search GA was applied, and to improve the solution from the first phase, a tabu search heuristic was used in the second phase. In addition, Karami and Hasanzadeh [31] suggested an HGA in which the heuristic algorithm produces the initial population, genetic operators deduct a solution, and the hill climbing method improves the solution. The summary of studies related to the one

Table 1. Summary of related papers.

	Methodology					Constraints				
	IP	Heuristic	Metaheuristic	Characteristics	Objective function	Room capacity	Multiple sessions	Periodicity	Consecutiveness	Remark
Lawrie [33]	√				None		√			
Akkoyunlu [4]		√			Minimize costs		√	√		
Mulvey [42]		√		Network-based optimization	Maximize efficiency	√				
Tripathy [48]		√		Lagrangean relaxation	Maximize efficiency	√	√			
Abramson [3]			√	Simulated annealing	Minimize conflicts of lectures		√			
Čangalović and Schreuder [9]			√	Graph colouring	Minimize time duration and free time		√		√	
Hertz [24]			√	Tabu search algorithm	Minimize conflicts of lectures	√				
Costa [11]			√	Tabu search algorithm	Minimize penalty		√		√	Building distances, lunch break
Erben and Keppler [21]			√	Genetic algorithm	Maximize people's satisfaction	√	√		√	
Deris *et al*. [17]		√		Constraint-based reasoning	Maximize people's satisfaction and utilization of resources	√	√			
Carrasco and Pato [10]			√	Multi-objective genetic algorithm	Minimize penalty		√		√	Lunch break, people's preferences
Socha *et al*. [46]			√	Ant colony algorithms	Minimize penalty	√				ITC2002(Post-enrolment based), Socha data sets
Daskalaki and Birbas [12]	√				Minimize costs of assigning lectures	√	√		√	Splitted consecutiveness
Daskalaki *et al*. [13]	√				Minimize costs of assigning lectures	√	√		√	Splitted consecutiveness

Lewis and Paechter [34]			√	Genetic algorithm	Minimize penalty	√				ITC2002(Post-enrollment based)
Lewis and Paechter [35]			√	Grouping genetic algorithm	Minimize students' inconvenience					Students' convenience
MirHassani [40]	√				Minimize penalty	√	√	√		
Innet and Nuntasen [25]			√	Genetic algorithm	Minimize penalty					
McMullan [39]			√	Great deluge algorithm	Minimize penalty	√				ITC2002(Post-enrollment based)
Schimmelpfeng and Helber [45]	√				Minimize penalty	√				Students' year of study, teachers' preferences
Abdullah and Turabieh [1]			√	Genetic algorithms	Minimize penalty	√				ITC2002(Post-enrollment based), Socha datasets
De Causmaecker *et al*. [14]			√	Local search and tabu search algorithm	Minimize penalty		√			
Turabieh *et al*. [49]			√	Electromagnetism-like mechanism with great deluge algorithm	Minimize penalty	√				ITC2002(Post-enrollment based)
Lü and Hao [37]			√	Adaptive tabu search algorithm	Minimize penalty	√				ITC2007 (Curriculum based)
Zhang *et al*. [52]		√		simulated annealing algorithm	Minimize penalty		√		√	
Jat and Yang [26]			√	Two-phase HGA: GSGA[a] and tabu search	Minimize penalty	√				ITC2002(Post-enrollment based)

(*Continued*).

Table 1. Continued.

	Methodology					Constraints				
	IP	Heuristic	Metaheuristic	Characteristics	Objective function	Room capacity	Multiple sessions	Periodicity	Consecutiveness	Remark
Abdullah *et al*. [2]			√	Electromagnetic-like mechanism and great deluge algorithm	Minimize penalty	√				ITC2002(Post-enrollment based), ITC2007 (Curriculum based)
Al-Betar and Khader [5]			√	Harmony search algorithm	Minimize penalty	√				ITC2002(Post-enrollment based), Socha datasets
Karami and Hasanzadeh [31]			√	HGA using hill climbing method	Minimize penalty	√				ITC2002(Post-enrollment based), Socha datasets
Lach and Lübbecke [32]	√				Minimize penalty	√	√		√	
Kalender *et al*. [27]			√	Iterative selection hyper-heuristic	Minimize penalty	√				Curriculum-based Course Timetabling Problem
Soria-Alcaraz *et al*. [47]			√	Iterated local search hyper-heuristic	Minimize penalty	√				ITC2002(Post-enrollment based), ITC2007 (Curriculum based)
Wahid and Hussin [50]			√	Harmony search algorithm	Minimize penalty	√				ITC2007 (Curriculum based)
This study	√	√	√	Heuristic, HGA	Minimize penalty	√	√	√	√	

Note: Minimize penalty: Minimize penalty cost of violating soft constraints.
[a]Guided search genetic algorithm.

we propose is shown in Table 1. We can see that several researchers have considered either the periodicity or consecutiveness of multi-session lectures. In contrast with those studies, we consider these two conditions simultaneously as well as a novel and practical decision variable. With such a variable, one multi-session lecture can be decided based on the periodicity or consecutives condition, which provides new managerial space.

The layer-building and the bottom deepest left with fill strategies have been widely used in studies on the 3DCPP. The layers are designed to divide the container into several blocks [8,38,43]. The bottom deepest left with fill strategy was developed to determine the packed items' positions in the container [28,30,41]. See Zhu and Lim [53] and Feng *et al.* [22] for surveys of these two strategies, respectively.

This paper is organized as follows: Section 2 presents an MILP model for solving the UTP. Section 3 describes the solution space as described through the 3DCPP. The procedure of the HGA is given in Section 4. In Section 5, results and analyses from computational experiments and the comparison of the performances between the HGA and a tabu search algorithm are provided. Section 6 presents the conclusions.

2. A mathematical model

In this section, we discuss the UTP studied in this paper and propose our mathematical model.

2.1 *Problem description*

We consider L lectures, and lecture l is comprised of session number m_l and student number n_l, $l = 1,2, \ldots, L$. Each lecture contains at least one session in one week and each session of any lecture occupies exactly one period. Typically, one lecture is given by one professor, who must repeat it as determined by the number of sessions scheduled in the same classroom every week. This arrangement helps prevent confusion among faculty members and students. The number of students attending each lecture is decided by student enrolments, and the classrooms have fixed, but different, sizes. Each multi-session lecture may have either consecutiveness or periodicity constraints. There are R classrooms, and classroom r is of capacity c_r, $r = 1, 2, \ldots, R$. The UTP consists of assigning L lectures into a timetable of D working days and P periods in each day. The timetable can be repeatedly used during the whole semester. Existing UTPs in most universities involve weekly timetabling. The objective is to minimize the total penalty cost for the soft constraints without violating the hard constraints. A lecture is defined as unassigned if any session of that lecture is not assigned. One lecture is assigned to a day-period-room slot when one session of that lecture is assigned there. Moreover, a group contains several lectures, and lectures from the same group may appeal to the same students. Therefore, we attempt to avoid scheduling overlapping lectures from the same group. If two such lectures are assigned to consecutive periods in the same day, we attempt to reduce the distance between the classrooms for those two lectures. The hard constraints (H1–H5) and soft constraints (S1–S4) are as follows:

H1. At most, one session can be assigned to a day-period-room slot.

H2. Each session of a lecture must be scheduled in a distinct day-period-room slot.

H3. The number of students enrolled in the lecture should not exceed the capacity of the target room.

H4. If a multi-session lecture has the periodicity constraint, sessions must be spread over time (i.e. not back-to-back) and should be held during the same period and in the same classroom. The minimum interval between each session is one day.

H5. If a multi-session lecture has the consecutiveness constraint, sessions must be offered in continuous periods in the same day and in the same classroom.

S1. The number of unassigned lectures should be minimized.

S2. For each lecture, the number of students attending the lecture should be close to the capacity of the target room that hosts the lecture.

S3. The overlapping of lectures from the same group should be minimized.

S4. The distance between the classrooms hosting lectures from the same group and given in consecutive period slots in the same day should be minimized.

H1–H3 are typical hard constraints in the UTP. H4 presents the periodicity constraint, which is set to give faculty members and students time to prepare and review the lectures. The UTP with $D = 5$ is the most commonly seen in the real world. In this case, for a lecture with three sessions under the periodicity constraint, the sessions can only be assigned to Monday, Wednesday, and Friday, while our model and algorithm are developed for general cases. H5 presents the consecutiveness constraint when some lectures must be given so that related components of the learning experience, such as laboratory and discussion sessions, are properly accommodated. S1 is used to assign as many lectures as possible. S2 means that the classrooms should be fully occupied. S3 seeks to minimize the conflict between lectures typically attended by the same students to provide students more choices for lectures. S4 means that students, who have different lectures in consecutive periods in the same day, need not traverse long distances on campus.

2.2 *Mixed integer program formulation*

In many real cases, administrators at universities decide whether multi-session lectures have either the consecutiveness or the periodicity constraint. We consider this issue in our model by using the (0,1) decision variable, z_l. Let T be the set of multi-session lectures. Lecture l has the periodicity constraint if $z_l = 1$; otherwise, $z_l = 0$, $l \in T$. Let $\mathrm{PER}_{\mathrm{PC}}$ and $\mathrm{PER}_{\mathrm{CC}}$ be the percentages of multi-session lectures with the periodicity constraint and the consecutiveness constraint in all multi-session lectures, respectively. The following notations explain the model:

Indices:

a	index for days, $d \in \{1, 2, \ldots, D\}$
p	index for periods, $p \in \{0, 1, \ldots, P+1\}$
r	index for classrooms, $r \in \{1, 2, \ldots, R\}$
l	index for lectures, $l \in \{1, 2, \ldots, L\}$

Parameters:

c_r	capacity of classroom r
n_l	maximum number of students in lecture l
m_l	number of sessions of lecture l
N	number of lecture groups with the potential to have common students
L_k	group k of lectures with the potential to have common students, $k = 1, 2, \ldots, N$
S_{ij}	distance between classrooms i and j
a	upper bound on $\mathrm{PER}_{\mathrm{PC}}$
b	upper bound on $\mathrm{PER}_{\mathrm{CC}}$
U	number of multi-session lectures

M	a big positive number
w_l^{s1}	unit penalty cost associated with constraint $S1$ for unscheduled lecture l
w^{s2}	unit penalty cost associated with constraint $S2$
w^{s3}	unit penalty cost associated with constraint $S3$
w^{s4}	unit penalty cost associated with constraint $S4$

Decision variables:

$X_{d,l,p,r}$	1: if one session of lecture l is assigned to classroom r in period p on day d; 0: otherwise.
$Y_{l,p,r}$	1: if lecture l is assigned to classroom r in period p; 0: otherwise.
z_l	1: if lecture l with multiple sessions has periodicity requirement; 0: otherwise
f_l^{s1}	1: if lecture l is not assigned; 0: otherwise.
$f_{d,p,r}^{s2}$	gap between the size of classroom r and the number of students of the lecture assigned to this classroom in period p on day d.
$f_{d,k,p}^{s3}$	number of overlapping lectures in period p on day d between lectures in L_k.
$f_{d,i,j,p}^{s4}$	distance between classrooms for lectures i and j assigned in periods p and $(p+1)$ on day d.

This MILP model for the UTP is formulated as follows:

$$\min \sum_{l=1}^{L} w_l^{s1} f_l^{s1} + w^{s2} \sum_{d=1}^{D}\sum_{p=1}^{P}\sum_{r=1}^{R} f_{d,p,r}^{s2} + w^{s3} \sum_{d=1}^{D}\sum_{p=1}^{P}\sum_{k=1}^{N} f_{d,k,p}^{s3} + w^{s4} \sum_{k=1}^{N}\sum_{d=1}^{D}\sum_{p=1}^{P}\sum_{i\in L_k}\sum_{j\in L_k} f_{d,i,j,p}^{s4}. \tag{1}$$

Subject to

$$\sum_{d=1}^{D}\sum_{p=1}^{P} X_{d,l,p,r} = 0 \quad \forall l, r \text{with } n_l > c_r, \tag{2}$$

$$\sum_{l=1}^{L} X_{d,l,p,r} \le 1 \quad \forall d, p, r, \tag{3}$$

$$\sum_{d=1}^{D}\sum_{p=1}^{P}\sum_{r=1}^{R} X_{d,l,p,r} = m_l(1 - f_l^{s1}) \quad \forall l, \tag{4}$$

$$\sum_{p=1}^{P}\sum_{r=1}^{R} Y_{l,p,r} = z_l \quad \forall l \in T, \tag{5}$$

$$\sum_{d=1}^{D} X_{d,l,p,r} + m_l(1 - Y_{l,p,r}) \ge m_l \quad \forall l \in T, p, r, \tag{6}$$

$$X_{dlpr} + X_{(d+1)lpr} \le 1, \quad \forall d \le (D-1), l \in T, p, r, \tag{7}$$

$$(m_l - 1)(X_{d,l,p,r} - X_{d,l,p+1,r}) - Mz_l \leq \sum_{j=p-m_l+1}^{p-1} X_{d,l,j,r} \quad \forall d, l \in T, m_l \leq p \leq P, r, \tag{8}$$

$$(m_l - 1)(X_{d,l,p,r} - X_{d,l,p-1,r}) - Mz_l \leq \sum_{j=p+1}^{p+m_l-1} X_{d,l,j,r} \quad \forall d, l \in T, 1 \leq p \leq P - m_l, r, \tag{9}$$

$$x_{d,l,0,r} = X_{d,l,(P+1),r} = 0, \quad \forall d, l, r, \tag{10}$$

$$\frac{\sum_{l\in T} z_l}{U} \leq a, \tag{11}$$

$$\frac{U - \sum_{l\in T} z_l}{U} \leq b, \tag{12}$$

$$f_l^{s1} = 0, 1 \quad \forall l, \tag{13}$$

$$f_{d,p,r}^{s2} \geq \sum_{l=1}^{L} [(c_r - n_l)X_{d,l,p,r}] \quad \forall d, p, r, \tag{14}$$

$$\begin{aligned} f_{d,k,p}^{s3} &\geq \sum_{r=1}^{R} \sum_{l\in L_k} X_{d,l,p,r} - 1 \quad \forall d, p, \text{ and } k = 1, 2, \ldots, N \\ f_{d,i,j,p}^{s4} &\geq \sum_{t_1=1}^{R} \sum_{t_2=1}^{R} [(X_{d,i,p,t_1} - X_{d,i,p+1,t_1} + X_{d,j,p+1,t_2} - X_{d,j,p,t_2} - 1)S_{t_1,t_2}], \end{aligned} \tag{15}$$

$$\forall d, p < P + 1, \text{and } i, j \in L_k, \tag{16}$$

$$f_{d,p,r}^{s2} \geq 0 \quad \forall d, p, r, \tag{17}$$

$$f_{d,k,p}^{s3} \geq 0 \quad \forall d, p, \text{ and } k = 1, 2, \ldots, N, \tag{18}$$

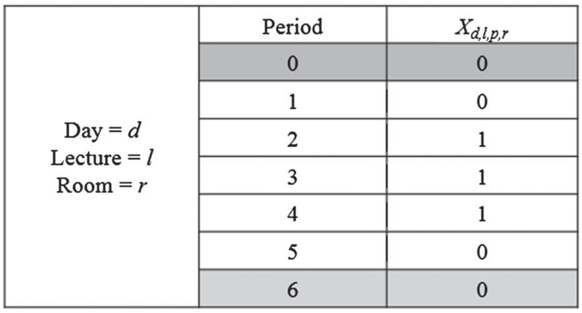
$$f_{d,i,j,p}^{s4} \geq 0 \quad \forall d, p, \text{ and } i, j \in L_k, \tag{19}$$

$$X_{d,l,p,r} = 0, 1 \quad \forall d, l, p, r, \tag{20}$$

$$Y_{l,p,r} = 0, 1 \quad \forall l, p, r, \tag{21}$$

$$z_l = 0, 1 \quad \forall l \in T. \tag{22}$$

The objective function (1) minimizes the total penalty from violations of soft constraints. Constraint (2) states that the size of a classroom should be greater than or equal to the number of students of the lecture assigned to that classroom. Constraint (3) indicates that, at most, one session can be assigned to a day-period-room slot. Constraint (4) ensures that f_l^{s1} is 0 if all sessions of lecture l are assigned. Constraints (5) and (6) ensure that multi-session lectures with periodicity constraints must be assigned to the same classroom in the same period. Constraint (7) prevents multi-session lectures with periodicity constraints from being assigned in successive days. Constraints (8) and (9) relate to the consecutiveness constraints: They show the difference between $X_{d,l,p,r}$ and $X_{d,l,p+1,r}$, and they reflect acceptable situations only when the difference between $X_{d,l,p,r}$ and $X_{d,l,p+1,r}$ is 1 and the sum of $X_{d,l,p,r}$ should be $(m_l - 1)$. Figure 1 illustrates an example of the assignment of a lecture under consecutiveness constraints. We can see that lecture l with three sessions is assigned to Periods 2 to 4, day d, and room r, $X_{d,l,2,r} - X_{d,l,1,r} = X_{d,l,4,r} - X_{d,l,5,r} = 1$. Then, from Constraints (8) and (9), we obtain $X_{d,l,3,r} + X_{d,l,4,r} = 2$ and $X_{d,l,2,r} + X_{d,l,3,r} = 2$, respectively, which means that the assignment of lecture l satisfies the consecutiveness constraint. Constraint (10) states that virtual periods with



	Period	$X_{d,l,p,r}$
Day = d Lecture = l Room = r	0	0
	1	0
	2	1
	3	1
	4	1
	5	0
	6	0

Figure 1. Example of an assigned lecture with three sessions under consecutiveness constraints.

a value of 0 should be added before the first period and after the last period. Constraints (11) and (12) ensure that the numbers of lectures with the respective consecutiveness and periodicity constraints should not exceed the upper bounds. In addition, $a + b \geq 1$ is established to guarantee the existence of a feasible z_l under these two constraints. Constraints (13) to (19) show the penalty variables for the soft constraints. In Constraint (15), $\sum_{r=1}^{R} \sum_{l \in L_k} X_{d,l,p,r}$ represents the total number of lectures in group k that are assigned to $\{d, p\}$. Then, $\sum_{d=1}^{D} \sum_{p=1}^{P} \sum_{k=1}^{N} f_{d,k,p}^{s3}$ represents the total number of overlapping lectures with the potential to appeal to the same students. Constraint (16) refers to the distance between classrooms for lectures i and j from the same group assigned to consecutive periods in the same day; if lectures from the same group have an overlapping timeslot, then the distance between classrooms is not considered because students should not select both of these lectures. From Constraint (16) we can see that, for lectures i and j from the same group, the distance between classrooms t_1 and t_2 generates penalty cost only when $X_{d,i,p,t1} - X_{d,i,p+1,t1} + X_{d,j,p+1,t2} - X_{d,j,p,t2} = 2$, which implies that $X_{d,i,p,t1} - X_{d,i,p+1,t1} = 1$ and $X_{d,j,p+1,t2} - X_{d,j,p,t2} = 1$. In this case, the session(s) of lecture i assigned to day d can be finished at period p and the session(s) of lecture j assigned to day d starts at period $p + 1$. Moreover, these two lectures do not have overlapping timeslots. Hence, Constraint (16) represents the case in which the penalty cost of S4 can be generated.

3. Solution space

This section presents the spatial solution based on the 3DCPP, which is used to maximize the utilization rate of the container packed with a set of cuboid items.

We can convert the UTP into the 3DCPP by considering lectures as items to pack into a container. We need to pack these items into a container with length of D, depth of R, and width of P under the hard and soft constraints. In particular, each multi-session lecture (e.g. lecture l) with the consecutiveness constraint can be considered as an item, the three dimensions of which are 1, 1, and m_l. We sort the classrooms in an ascending order of room sizes. Then, we define Classroom 1 as the smallest and Classroom R as the largest. The room axis is numbered with the index of classrooms (Figure 2).

3.1 *Layer-based bottom deepest left with fill strategy*

The layer-based bottom deepest left with fill (LBDLF) strategy is used to assign the lectures into the classrooms effectively. For one lecture to be assigned, we develop a layer in the horizontal to mark the smallest classroom that can accommodate the number of students assigned to it.

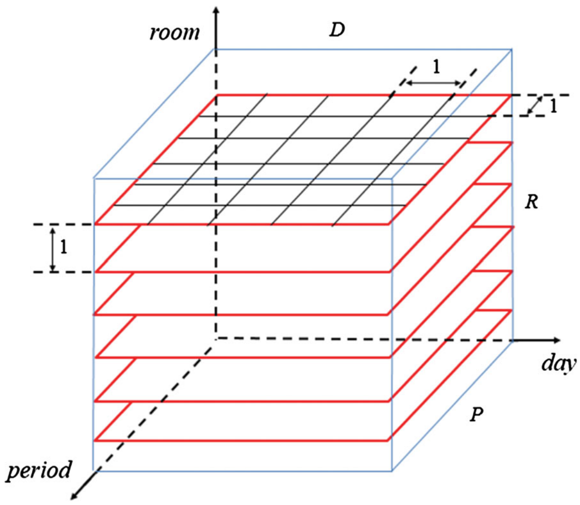


Figure 2. Graphical presentation of solution space with 5 days, 6 periods each day, and 7 classrooms.

Considering $S2$, we give the higher priority to the smallest classroom above the base layer. In that classroom, the timeslots must be examined from the left upper corner. It means that we attempt to assign lectures to the feasible timeslot(s) with the smallest values of day and period. Figure 3 illustrates the pseudo-code for the LBDLF strategy.

As Figure 3 shows, we choose the first slot for lecture l by using the LBDLF strategy. If lecture l is a single-session lecture or if it is a multi-session lecture with the consecutiveness constraint, then the complete assignment of lecture l is determined for the first slot. If lecture l is a multi-session lecture with the periodicity constraint, then we will try the assignment from the first slot and other slots which satisfy H4.

```
Start
Input R: List of classrooms in descending order of size
Input L: List of lectures
for $l$=1 to $|L|$ do
  Create a horizontal layer horizontal at the smallest classroom
  (e.g., classroom $i$), the size of which is no smaller than $n_l$
for $r=i$ to $|R|$ do
  Input $VS_r$: List of vacant slots in classroom $r$
  sort List $VS_r$ according to deepest-left order;
  for $j$=1 to $|VS_r|$ do
    if all classes of lecture $l$ can be assigned from slot $j$ then
      assign lecture $l$ from slot $j$ in classroom $r$;
      break;
    end if
  end for
  if lecture $l$ is assigned then
    break;
  end if
end for
end for
End.
```

Figure 3. Pseudo-code for the LBDLF strategy.

3.2 *Swap neighbourhoods*

For a solution, we can swap the assignments of two lectures as if we are swapping the positions of two solid items in one container. According to De Causmaecker *et al*. [14], we discuss three types of swap:

With the *horizontal swap*, lectures assigned in the same classroom can be swapped; with the *vertical swap*, lectures assigned in the same day-period timeslot can be swapped; with the *mixed swap*, the horizontal and vertical swaps are combined. In a mixed swap, both the classrooms and day-period timeslots of two lectures will be changed.

The horizontal and mixed swaps may decrease the penalty costs caused by violating S3. Because the vertical swap does not change the timeslots of the lectures, it has no impact on the penalty cost associated with S3. It is easy to see that these three types of swaps may decrease the penalty costs caused by violating S4. However, they achieve these improvements through different mechanisms. The horizontal swap may reduce S4 violations by not assigning two lectures in consecutive periods of the same day, and the vertical swap may decrease the distance between the two classrooms. However, the penalty costs from violating S1 and S2 cannot be decreased under the three types of swap. We avoid swaps that could not reduce penalty costs.

4. Hybrid GA

This section presents the HGA applied toward the UTP as per the GA that includes the LBDLF strategy and a local algorithm.

In the HGA, the sequence of assigning lectures and the conditions of consecutiveness and periodicity constraints of multi-session lectures are represented by a chromosome. The fitness of each chromosome is determined by objective function (1). For a chromosome, an initial assignment solution can be obtained based on the LBDLF. Then, the solution is improved through a local search. Figure 4 shows the flow chart of HGA creation.

4.1 *Local search*

The local search is developed to improve the initial solution obtained by each chromosome. We compare all pairs of assigned lectures. If one pair of lectures generates a penalty cost from violation of either $S3$ or $S4$, then we swap one of the lectures with another in the neighbourhood. We do not use the vertical swap if the pair of lectures does not generate a penalty cost from violation of $S4$. For each pair of lectures, the local search is run for a number of iterations, and we save the swap generating the smallest penalty cost.

4.2 *Chromosome representation*

A chromosome in the HGA has two rows with L columns. Therefore, each chromosome contains L genes. The first row presents the sequence of assigned lectures and the second one presents the consecutiveness and periodicity constraints of the lectures. We let r_l represent the consecutiveness and periodicity constraints of lecture l and r_l can be equal to 0, 1, and 2. If $r_l = 0$, then lecture l does not have the consecutiveness or periodicity constraint. If $r_l = 1$, then lecture l has the consecutiveness constraint. If $r_l = 2$, then lecture l has the periodicity constraint. Figure 5 shows an example of the chromosome structure. We first assign Lecture 2 to a vacant slot based on the LBDLF strategy, and then Lecture 6 is assigned, and the process is repeated in sequence

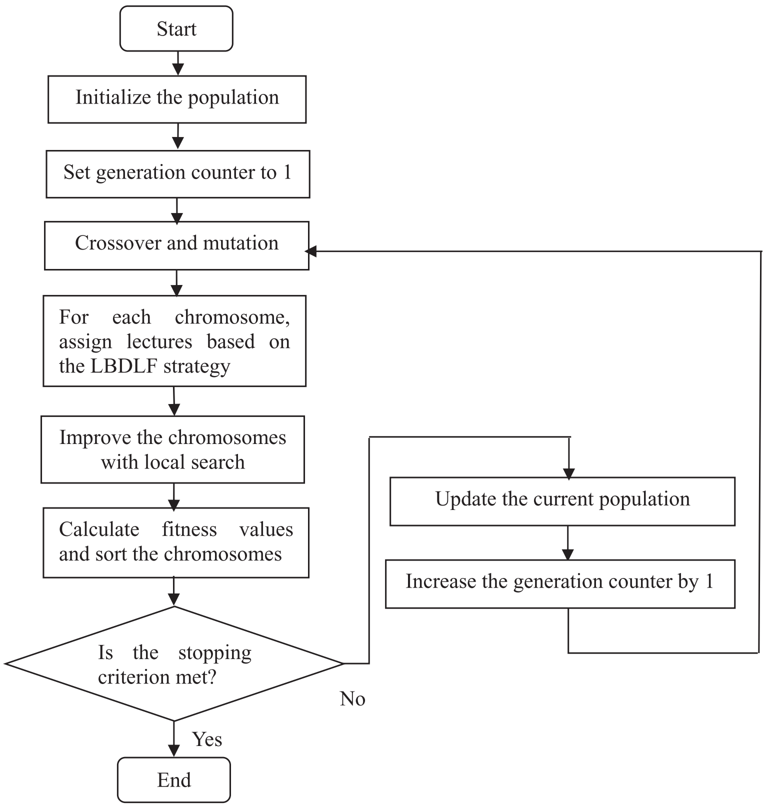

Start
Initialize the population
Set generation counter to 1
Crossover and mutation
For each chromosome, assign lectures based on the LBDLF strategy
Improve the chromosomes with local search
Calculate fitness values and sort the chromosomes
Is the stopping criterion met?
No
Update the current population
Increase the generation counter by 1
Yes
End

Figure 4. Flowchart of the HGA.

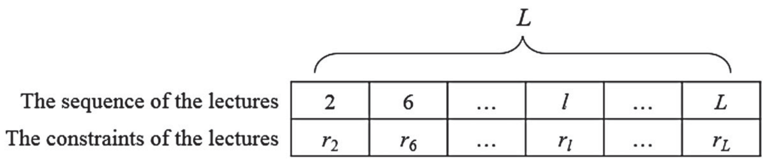


Figure 5. Chromosome structure.

such that the lecture genes are assigned to specific slots. We can obtain a complete solution as depicted by a chromosome and through use of the LBDLF strategy.

Equation (23) is used to measure the fitness value of chromosome i, $i = 1,2, \ldots,$ POP, where POP represents the population size.

$$\text{Fitness}_i = 1/\text{the total penalty cost in chromosome } i. \tag{23}$$

4.3 *Population initialization*

Each gene on the chromosome indicates a lecture, and the sequence of the genes shows the order of assigned lectures into classrooms. Throughout the procedure, the directed initialization and random initialization methods were applied to chromosomes. For the directed method, one special chromosome initiates the process such that multi-session lectures are assigned before the single-session lectures. Furthermore, among multi-session lectures, those with periodicity constraints are scheduled before the lectures with consecutiveness constraints, and they are assigned according to the descending order of the required periods in a week. However, the remaining genes on the chromosome are randomly arranged as per the Fisher-Yates shuttle method [23]. The use of two different methods helps the researcher achieve two goals for finding solutions: The directed initialization method provides fast convergence to a local optimum and the random initialization method maintain the genetic diversity of the initial population.

4.4 *Chromosome decoding*

The lectures are assigned in sequence by the LBDLF strategy. The number of lecture students is used to generate a layer that reveals the number of feasible classrooms. The list of classrooms is prepared in descending order by capacity, which must be no smaller than the number of students attending the lecture. Sometimes the most suitable classroom has already been scheduled for other lectures. In this case, a classroom one size larger than the best fitted classroom is considered for assigning the lecture. Following the LBDLF strategy, the starting point is always the first period on Monday. The objective is to minimize the total penalty cost, so the chromosome with the largest fitness value is considered the best one. Elitism strategy is used to select good chromosomes that transmit the best genes and help maintain the diversity of the population. Also, the use of this strategy can prevent the disappearance of good solutions during the process of crossover and mutation [15]. After each original chromosome population was evaluated, the one with the largest fitness value was saved.

4.5 *Chromosome selection, crossover, and mutation*

In the HGA, a large perturbation in the chromosome helps reveal the best solution, because the HGA, contrary to a simple GA, has a local optimization property. Order crossover and inversion mutation helps maintain chromosome diversity among the population. In each generation of the HGA, a certain number of chromosome pairs among the current population are selected for crossover and mutation. Each pair of chromosomes can be obtained with the following approach. Without loss of generality, let these two chromosomes in one pair be P_1 and P_2. Let the two new chromosomes generated from the crossover and mutation by P_1 and P_2 be O_1 and O_2, respectively. First, two chromosomes were randomly chosen from the current population which are denoted as K_1 and K_2, respectively. Second, of K_1 and K_2, the one with the higher fitness value is chosen as P_1. Third, P_2 is selected through the first two steps.

For each pair of chromosomes, crossover occurs with a probability of P_c. During a crossover, two arbitrary cutting lines, $c1$ and $c2$, are placed on P_1. The genes between two cutting lines are copied and placed in the corresponding location on O_1. The genes on O_1 following the second cutting line are replaced with those copied in the order of sequence from the corresponding space on P_2. O_2 is built through the same process, except the roles of P_1 and P_2 are reversed.

We conduct a two-step mutation for O_1 and O_2 to avoid searching in local optimal solutions. In the first step, two genes of O_1 are randomly selected and swapped with a probability that is denoted as P_{m1}. In the second step, the consecutiveness and periodicity conditions of each genes representing multi-session lectures are randomly changed with a probability that is denoted as P_{m2}. These two steps are also conducted for O_2. Then, O_1 and O_2 are added in the current population.

4.6 *Updating the population and stopping criteria*

The chromosomes are listed in descending order of fitness value and only the top 50% is included in the new population. This method is widely used, in general, for termination of GAs. However, quickly reaching a local optimum should be avoided to prevent wasteful iterations of genetic operations. The diversity index (div) is used to prevent reaching local optimums too quickly. The fitness value of each chromosome is evaluated after the genetic operation step for all the chromosomes in one generation. The best chromosome, f_{best}, is selected by comparing the fitness values of each chromosome. Then, the average of the fitness values among the population in one generation can be calculated as f_{avg}. The value of div is calculated as Equation (24). If the index div converges to zero, the worst chromosomes are replaced by new randomly initialized chromosomes. Thus, the new generation can maintain the diversity of the population.

$$\text{div} = \frac{f_{\text{best}} - f_{\text{avg}}}{f_{\text{avg}}}. \tag{24}$$

The fitness value of the best chromosome identified in the preceding step is compared to that of the best chromosome in the new population. If the fitness value of the best chromosome in the original population is less than the fitness value of new one, the new one is chosen as per the elitism selection strategy.

The stopping criterion is met when the generation counter reaches the maximum number of generations. If the stopping criterion is not met, we continue running the HGA by creating a new generation.

5. Computational experiments

This section shows our three computational experiments. In Experiment 1, we run the HGA on the instances generated by our rule. The effect of PER_{CC} and PER_{PC} are discussed in Experiment 2, which is based on the same instances analysed in Experiment 1. For Experiment 3, we run the algorithm on new instances generated based on the well-known benchmarks of the ITC2007. Because our UTP is different from the one involved in the ITC2007 in several respects, we generated new instances by incorporating several ITC2007 benchmarks, such as the total number of sessions, classroom capacity, and student number, into the instances analysed. The instances we analysed and the HGA source code are publicly available at http://scm.snu.ac.kr/publication/code/GA_Instances.rar.

The MILP model was coded in FICO Xpress-IVE version 7.3, and the HGA models were run in the Java programming language in Windows 7 on a PC with Intel Core 3.6 GHz. Two sets of experiments were conducted in this study. Because the tabu search has been used in many existing studies on the UTP, we compare our HGA with a tabu search algorithm based on De Causmaecker *et al*. [14]. By searching the horizontal, vertical, and mixed swap neighbourhoods, we swap the lecture assigned to each timeslot with other lectures. We switch from one neighbourhood to another one when the swaps reach a point where they are not improving the output; we set the search limit per neighbourhood to 10 based on results from pilot tests. The tabu search for one timeslot stops when we finish searching the three neighbourhoods. To make the tabu search algorithm more competitive, we add the initial solution generation mechanism that Lü and Hao [37] have proven efficient. Under this mechanism, the consecutiveness and periodicity constraints for each multi-session lecture are determined to maximize the number of available periods for the lecture.

5.1 *Setting parameters for experiment 1*

We choose the UTP used in most South Korean universities as an example for our experiments. In the authors' university, a weekly timetabling with five working days is determined, and the timetable is used every week during the entire 16-week semester. In the timetable for each classroom, the lecture timeslots consist of six 90-minute periods per day from Monday through Friday. Lectures should be assigned to timeslots, and each timeslot can be defined as a set consisting of a day, period, and classroom.

Prior to assigning lectures to the timeslots, schedulers need to decide the periodicity and consecutiveness conditions for multi-session lectures. We tested our HGA and the tabu search on small, medium, and large problems. In those problems, the classroom capacity follows a uniform distribution in [50, 200]. The distance between every pair of classrooms follows a uniform distribution in [0, 10]. The maximum number of students in each lecture follows a uniform distribution in [20, 120]. We arbitrarily selected four values, 30, 0.1, 20, and 5, as the unit penalty costs for violating S1, S2, S3, and S4, respectively. We decided that possibilities for a lecture to have one, two, and three sessions are 10%, 80%, and 10%, respectively, which indicates that the two-sessions-per-week format is the most common. The lectures that require one or three periods a week often have special features, such as association with laboratory classes, seminars, and the like. Most universities in China also have this type of lecture scheme. Because each university may assign different values to the period and time length, our algorithm can be easily customized.

To solve the problem efficiently and to set the parameters for the HGA, we conducted pilot tests with large problems. The results are shown in Figures 6–10. Population sizes of 50, 150, 200, 250, and 300 were tested, and the results show that the best population size according to the objective function and optimal computation time is 200. When the population is more than 200, the best solution obtained from the HGA cannot be improved even if the computation time is increased. For the same reason, we found that the HGA has the highest efficiency when the maximum generation number is set to 400. The results show that 0.7 is the best value for P_c as determined by the mean of the objective function value. The impacts of P_{m1} and P_{m2} on the objective function value are shown in Figures 9 and 10. We set P_{m1} and P_{m2} to 0.2 and 0.1, respectively. In addition, based on the pilot tests, we set the number of iterations of the local search in the HGA as 100 for the experiments.

5.2 *Experiment 1*

Experiment 1 was conducted to compare the performances of the tabu search and the HGA. Randomly generated problem sets were categorized into small, medium, and large. The small

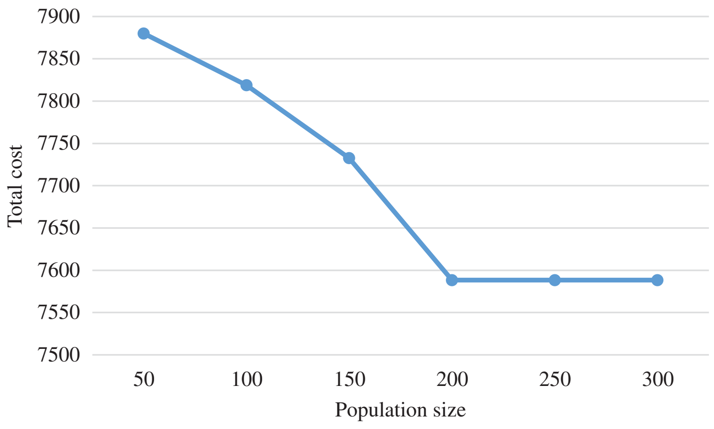


Figure 6. Total costs under different values of population size.

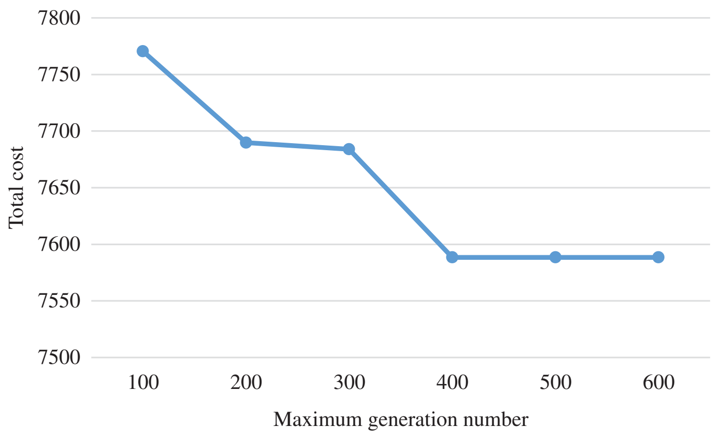


Figure 7. Total costs under different values of maximum generation number.

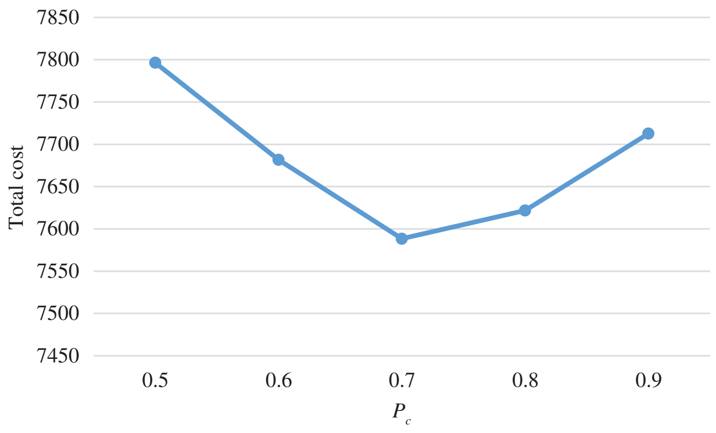


Figure 8. Total costs under different values of P_c.

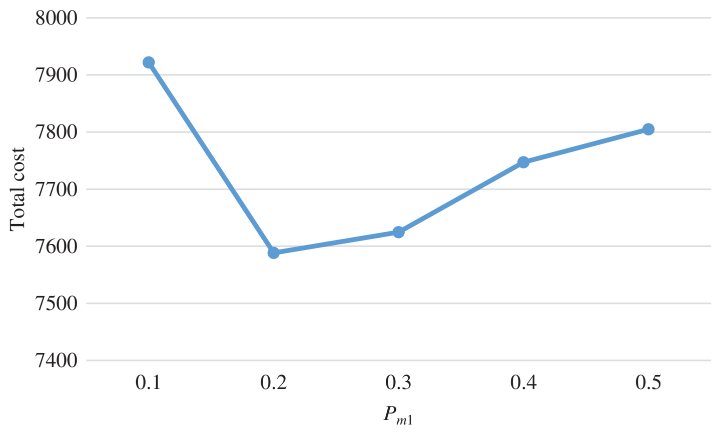


Figure 9. Total costs under different values of P_{m1}.

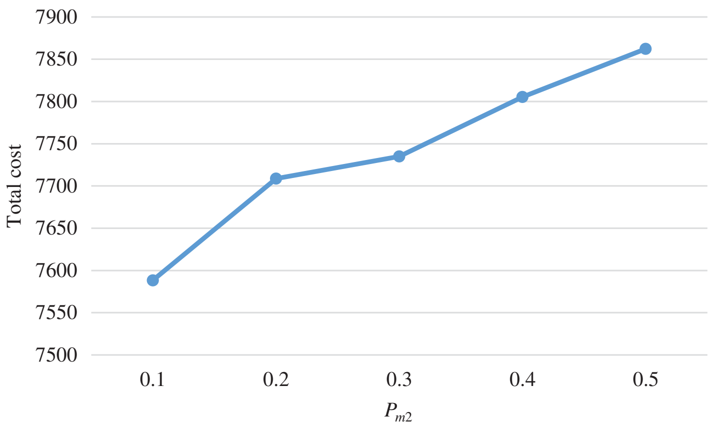


Figure 10. Total costs under different values of P_{m2}.

problems were limited to 5 classrooms, 100 lectures, and 4 groups; the medium problems were limited to 10 classrooms, 200 lectures, and 4 groups; and the large problems were limited to 15 classrooms, 300 lectures, and 6 groups. For each problem size, we generated five instances. For each of these instances, we randomly assigned a lecture to a group, and the lecture could not be assigned to any group with a possibility of 40%. Table 2 summarizes the values of U, N, and $|L_k|$ for each instance, $k = 1,2, \ldots, N$. The symbol '/' represents no value for that parameter.

Tables 3–5 show the comparisons of the tabu search and the HGA for the instances of different problem sizes. In these problems, PER_{PC} and PER_{CC} were not limited. As can be seen from the results, the HGA can obtain better solutions than the tabu search within an acceptable time. As

Table 2. Values of U, N, and $|L_k|$ for the instances.

	Instance of small size problem					Instance of medium size problem					Instance of large size problem				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
U	88	87	86	77	88	166	179	179	163	176	264	262	250	244	261
N	4	4	4	4	4	4	4	4	4	4	6	6	6	6	6
L_1	8	11	14	5	9	22	20	21	18	22	25	32	26	27	22
L_2	10	12	13	8	18	17	26	9	24	27	26	34	27	31	27
L_3	14	11	9	7	8	25	26	24	23	24	38	31	27	19	25
L_4	13	15	15	16	16	20	29	38	27	28	29	34	36	34	36
L_5	/	/	/	/	/	/	/	/	/	/	37	41	27	29	39
L_6	/	/	/	/	/	/	/	/	/	/	15	14	16	15	23

Table 3. Comparison of the tabu search and the HGA for instances with 100 lectures, 5 classrooms, and 4 groups.

	tabu search		HGA	
Instance	Objective value	Computation time (seconds)	Objective value	Computation time (seconds)
1	1794.5	0.53	1685.6	14.24
2	1963.8	0.52	1786.3	17.22
3	2105.8	0.53	1572.0	16.50
4	1973.6	0.54	1322.5	17.04
5	2282.9	0.70	1777.0	16.74

Table 4. Comparison of the tabu search and the HGA for instances with 200 lectures, 10 classrooms, and 4 groups.

	tabu search		HGA	
Instance	Objective value	Computation time (seconds)	Objective value	Computation time (seconds)
1	4658.6	3.41	3802.0	32.86
2	5493.9	3.53	5118.8	35.27
3	4302.0	3.07	3949.0	34.42
4	5235.8	3.33	4501.6	35.57
5	4265.4	3.25	4017.2	37.02

Table 5. Comparison of the tabu search and the HGA for instances with 300 lectures, 15 classrooms, and 6 groups.

	tabu search		HGA	
Instance	Objective value	Computation time (seconds)	Objective value	Computation time (seconds)
1	7856.8	11.43	7588.4	58.98
2	7840.2	12.01	7673.0	58.32
3	8788.5	11.43	7870.0	61.55
4	9015.0	12.11	7814.6	57.97
5	8151.1	11.75	7551.4	62.87

Table 6. One solution of Instance 1 from Table 3.

	Day 1					Day 2					Day 3					Day 4					Day 5				
	Classroom					Classroom					Classroom					Classroom					Classroom				
Period	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	*65*	*16*	*75*	*29*	*38*	*85*	*27*	*41*	*53*	*8*	*65*	*16*	*35*	*29*	*38*	*13*	*37*	*41*	*53*	*8*	*77*	*54*	*35*	*29*	*38*
2	*55*	*22*	*4*	*1*	*15*	*85*	*27*	*89*	*52*	*87*	*55*	*50*	*24*	*1*	*15*	*18*	*37*	*89*	*52*	*74*	*14*	*50*	*24*	*1*	*96*
3	*88*	*22*	*4*	*12*	*48*	*17*	*42*	*21*	*56*	*87*	*11*	*33*	*81*	*12*	*48*	*17*	*42*	*84*	*56*	*74*	*11*	*33*	*10*	*64*	*96*
4	*98*	*63*	*51*	*46*	*36*	*90*	*23*	*21*	*57*	*70*	*80*	*34*	*81*	*46*	*36*	*19*	*47*	*84*	*59*	*70*	*80*	*68*	*10*	*64*	*31*
5	*98*	*63*	*51*	*72*	*26*	*90*	*23*	*21*	*57*	*73*	*39*	*34*	*81*	*72*	*26*	*19*	*47*	*84*	*59*	*73*	*20*	*68*	*91*	*62*	*31*
6	*98*	*25*	*51*	*71*	*93*	*92*	*49*	*5*	*58*	*69*	*6*	*25*	*43*	*71*	*93*	*92*	*49*	*5*	*58*	*69*	*20*	*30*	*43*	*32*	*31*

Note: The number in italic represents the lecture number.

the example, Table 6 shows one assignment solution of Instance 1 in Table 3 obtained from the HGA.

Next, we analysed the percentage of assigned multi-session lectures in all of the multi-session lectures. For each size of problem, when the number of classrooms is increased to a certain value, most multi-session lectures can be assigned (with a percentage greater than 96%). Moreover, such values do not correlated linearly with the number of lectures. The number of multi-session lectures in small problems are approximately twice as big as the number in medium problems. However, the numbers of required classrooms necessary to assign most of the multi-session lectures in the small and medium problems are 7 and 13, respectively. Similar phenomena can be found in the relationship between medium and large problems. Figure 11 illustrates the average percentages of assigned multi-session lectures for the five instances of small, medium, and large problems. We also conducted a sensitivity analysis to discuss the trend of overlap under different values of w^{s3}. Based on the sensitivity analysis, we found that the number of overlapping lectures is decreased as w^{s3} is increased. For instance, in small problems, the average value of overlapping

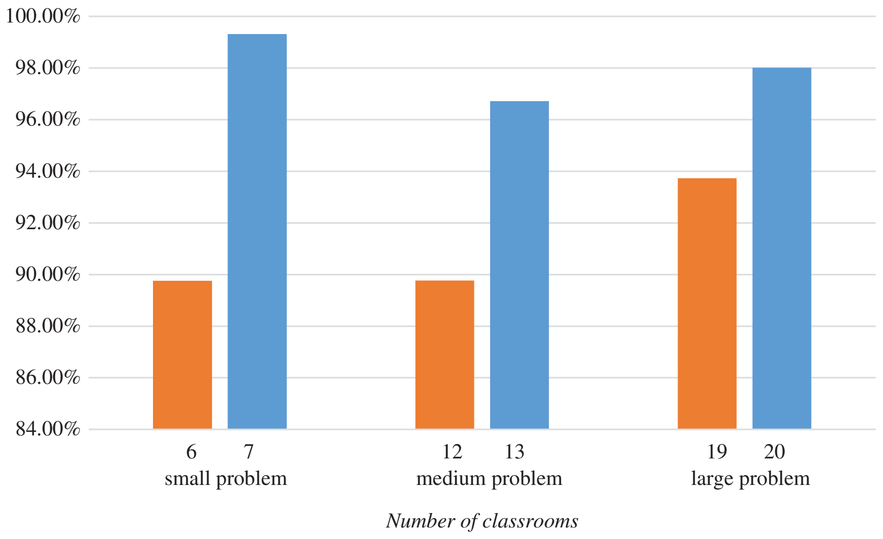


Figure 11. The percentages of assigned multi-lectures under different numbers of classrooms.

lectures in the five instances is decreased from 14.0 to 6.6, when the value of w^{s3} is increased from 10 to 50.

5.3 *Experiment 2*

We conducted Experiment 2 to examine the effect of increasing $\mathrm{PER_{CC}}$ when $\mathrm{PER_{PC}} + \mathrm{PER_{CC}} = 1$. For each instance, we ran the HGA five times and obtained the average value of the solutions. Figure 12 shows the results from the five instances of small problems. We can see that the HGA obtains the best solutions with the lowest costs when $\mathrm{PER_{CC}}$ was limited within the range of 70–80% for all five instances. Figure 13 illustrates the results from the five instances of medium problems; the best solutions were obtained when the $\mathrm{PER_{CC}}$ was limited within the range of 40% and 80%. In the five instances obtained from large problems, the best solutions were obtained when the $\mathrm{PER_{CC}}$ was limited within the range of 60% and 80% as Figure 14 shows.

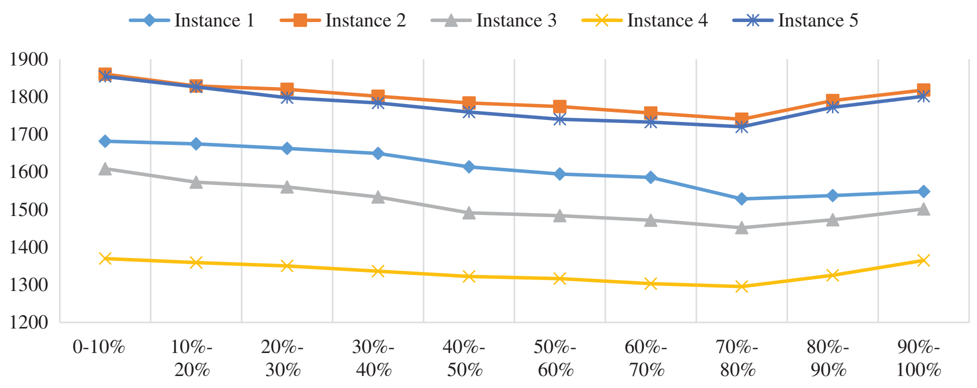


Figure 12. Total costs under different ranges of $\mathrm{PER_{CC}}$ for small problems.

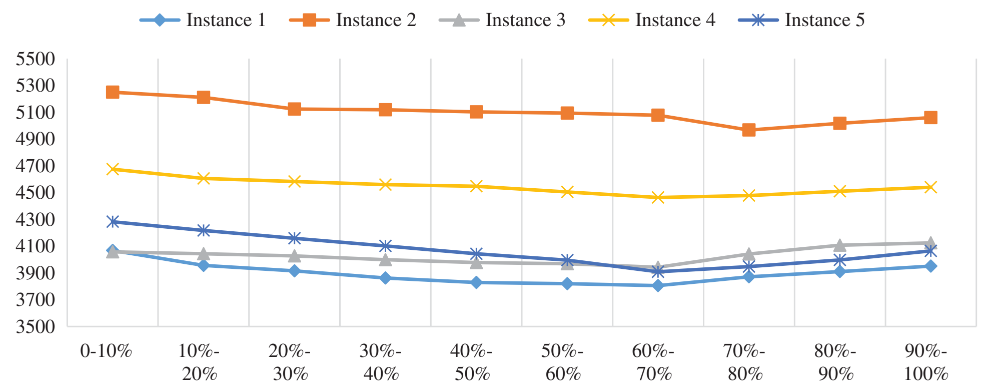


Figure 13. Total costs under different ranges of PER_{CC} for medium problems.

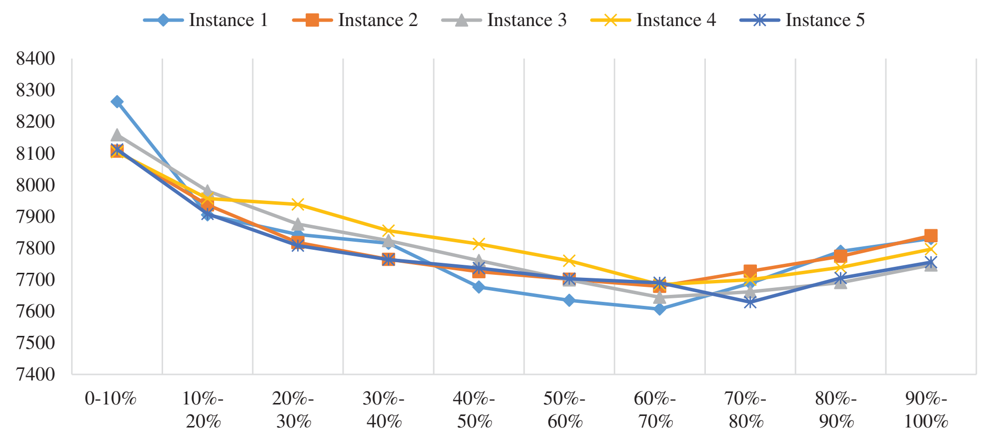


Figure 14. Total costs under different ranges of PER_{CC} for large problems.

The PER_{CC} and PER_{PC} have significant impacts on the optimal solutions. Specifically, these experiments show that administrators can improve the quality of their schedule by deciding the values of PER_{PC} and PER_{CC}. When the problem is large, they have more options for assigning the multi-session lectures because the optimal PER_{CC} and PER_{PC} values are found within a wider range, as Figures 13 and 14 show.

5.4 *Experiment 3*

We conducted Experiment 3 to test our HGA on the instances based on the well-known benchmarks used in ITC2007. We applied several characteristics of the benchmarks into our instances. For example, the number of sessions in each of our new instances is the same as the number of lectures in the corresponding benchmarks. The benchmark number of classrooms, room capacity, and building information are also retained in our instances. The results in the new experiments show that our HGA works well with the benchmark-adjusted instances. We use the same penalty costs and the same ratio of multi-session lectures as we used in Experiment 1. Based on pilot tests, we set the maximum generation number and population size of the HGA to 100 and 150, respectively. Other parameters of the HGA are the same as those in Experiment 1. Table 7 shows the results obtained from the tabu search and the HGA for these instances. We can see that the HGA can obtain better solutions than the tabu search within an acceptable time.

Table 7. Comparison of the tabu search and the HGA for instances generated based on the benchmarks in the ITC2007.

	tabu search		HGA	
Instance	Objective value	Computation time (seconds)	Objective value	Computation time (seconds)
Mcom 1	1025.6	0.524	1010.4	57.602
Mcom 2	3974.9	4.373	3473.9	57.888
Mcom 3	2531.7	2.952	2314.7	49.124
Mcom 4	4905.3	3.977	4477.4	72.127
Mcom 5	2715.2	0.548	2216.0	34.336
Mcom 6	5510.5	5.700	5205.4	85.402
Mcom 7	5543.5	9.636	5456.2	77.736
Mcom 8	5208.2	7.448	5184.5	73.465
Mcom 9	3818.9	4.406	3750.9	63.341
Mcom 10	4862.5	7.583	4680.3	83.816
Mcom 11	921.0	0.441	671.6	35.780
Mcom 12	2028.3	1.652	1602.1	48.306
Mcom 13	5641.5	5.179	4786.1	59.891
Mcom 14	3604.8	3.658	3451.3	63.281
Mcom 15	3773.3	2.900	3417.9	50.281
Mcom 16	6145.3	7.361	6108.1	67.310
Mcom 17	3637.9	5.476	3599.5	71.849
Mcom 18	1201.6	0.588	967.0	33.886
Mcom 19	3994.5	3.531	3901.1	63.969
Mcom 20	5280.2	7.028	5073.1	82.580
Mcom 21	3296.1	5.778	2657.3	75.369

6. Conclusions

According to the realistic conditions observed in many eastern Asian universities, we set the consecutiveness and periodicity conditions of multi-session lectures as novel decision variables. Decision makers can obtain the optimal solution by assigning the consecutiveness and periodicity constraints to the multi-session lectures within a defined ratio. Numerical experiments showed that such ratios influence the solutions obtained from the algorithm significantly.

This paper presented an MILP model and an HGA to solve the UTP with consecutiveness and periodicity constraints. We designed the HGA based on the 3DCPP. The LBDLF strategy is used to assign the lectures into the classrooms effectively. An efficient local search algorithm was proposed to improve the solution determined by the chromosomes and the LBDLF strategy. By transforming the UTP into a 3DCPP, we can obtain more options for designing efficient algorithms. We compared our HGA with the tabu search algorithm, which has been proven efficient in solving the UTP. The results showed that our HGA can yield better solutions within an acceptable time than the tabu search for small, medium, and large problems.

The MILP and the HGA featured in this paper can easily be extended to problems with more features and constraints. Some other hard and soft constraints can be added into the MILP, such as the preferences of professors and avoidance of overlapping mandatory lectures, which will complicate the problem. Researchers may attempt other algorithm designs using the 3DCPP ideas and other relevant problems to develop efficient algorithms for the UTP for future research.

Acknowledgments

The authors are grateful to valuable comments by associate editor and three referees to improve earlier versions of the paper.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This research was supported by the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning [grant number 2015R1A2A1A15053948].

ORCiD

Ilkyeong Moon http://orcid.org/0000-0002-7072-1351

References

[1] S. Abdullah and H. Turabieh, *Generating university course timetable using genetic algorithms and local search*, Proceedings of the Third International Conference on Convergence and Hybrid Information Technology, 2008, pp. 254–260.

[2] S. Abdullah, H. Turabieh, B. McCollum, and P. McMullan, *A hybrid metaheuristic approach to the university course timetabling problem*, J. Heuristics 18(1) (2012), pp. 1–23.

[3] D. Abramson, *Constructing school timetables using simulated annealing: Sequential and parallel algorithms*, Manage. Sci. 37(1) (1991), pp. 98–113.

[4] E. Akkoyunlu, *A linear algorithm for computing the optimum university timetable*, Comput. J. 16(4), (1973), pp. 347–350.

[5] M.A. Al-Betar and A.T. Khader, *A harmony search algorithm for university course timetabling*, Ann. Oper. Res. 194(1), (2012), pp. 3–31.

[6] H. Babaei, J. Karimpour, and A. Hadidi, *A survey of approaches for university course timetabling problem*, Comput. Ind. Eng. 86 (2015), pp. 43–59.

[7] R. Bellio, S. Ceschia, L.D. Gaspero, A. Schaerf, and T. Urli, *Feature-based tuning of simulated annealing applied to the curriculum-based course time tabling problem*, Comput. Oper. Res. 65 (2016), pp. 83–92.

[8] A. Bortfeldt and H. Gehring, *A hybrid genetic algorithm for the container loading problem*, European J. Oper. Res. 131 (2001), pp. 143–161.

[9] M. Čangalović and J.A. Schreuder, *Exact colouring algorithm for weighted graphs applied to timetabling problems with lectures of different lengths*, European J. Oper. Res. 51(2), (1991), pp. 248–258.

[10] M.P. Carrasco and M.V. Pato, *A multiobjective genetic algorithm for the class/teacher timetabling problem*, in *Practice and Theory of Automated Timetabling III*, E. Burke and W. Erben, eds., Springer, Berlin, 2001, pp. 3–17.

[11] D. Costa, *A tabu search algorithm for computing an operational timetable*, European J. Oper. Res. 76(1), (1994) pp. 98–110.

[12] S. Daskalaki and T. Birbas, *Efficient solutions for a university timetabling problem through integer programming*, European J. Oper. Res. 160 (2005), pp. 106–120.

[13] S. Daskalaki, T. Birbas, and E. Housos, *An integer programming formulation for a case study in university timetabling*, European J. Oper. Res. 153(1), (2004), pp. 117–135.

[14] P. De Causmaecker, P. Demeester, and G.V. Berghe, *A decomposed metaheuristic approach for a real-world university timetabling problem*, European J. Oper. Res. 195 (2009), pp. 307–318.

[15] K.A. De Jong, *An analysis of the behavior of a class of genetic adaptive systems*, Doctoral Dissertation, University of Michigan, 1975.

[16] D. de Werra, *An introduction to timetabling*, European J. Oper. Res. 19(2) (1985), pp. 151–162.

[17] S. Deris, S. Omatu, H. Ohta, and P. Samat, *University timetabling by constraint-based reasoning: A case study*, J. Oper. Res. Soc. 48(12), (1997), pp. 1178–1190.

[18] P. Dige, C. Lund, and H.F. Ravn, *Timetabling by simulated annealing*, in *Applied Simulated Annealing*, R.V.V. Vidal, ed., Springer, Berlin, 1993, pp. 151–174.

[19] K.A. Dowsland, *A timetabling problem in which clashes are inevitable*, J. Oper. Res. Soc. 41(10), (1990), pp. 907–918.

[20] A. Drexl and F. Salewski, *Distribution requirements and compactness constraints in school timetabling*, European J. Oper. Res. 102(1), (1997), pp. 193–214.

[21] W. Erben and J. Keppler, *A genetic algorithm solving a weekly course-timetabling problem*, in *Practice and Theory of Automated Timetabling*, E. Burke and P. Ross, eds., Springer, Berlin, 2005, pp. 198–211.

[22] X.H. Feng, I.K. Moon and J.H. Shin, *Hybrid genetic algorithms for the three-dimensional multiple container packing problem*, Flexible Serv. Manuf. J. 27 (2015), pp. 451–477.

[23] R.A. Fisher and F. Yates, *Statistical tables for biological, agricultural and medical research*, in *Statistical Tables for Biological, Agricultural and Medical Research*, Ed. 3., Oliver and Boyd, London, 1949, pp. 1–112.

[24] A. Hertz, *Tabu search for large scale timetabling problems*. European J. Oper. Res. 54(1) (1991), pp. 39–47.

[25] S. Innet and N. Nuntasen, *University timetabling using evolutionary computation*. WSEAS Trans. Adv. Eng. Educ. 12 (2007), pp. 243–250.

[26] S.N. Jat and S. Yang, *A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling*, J. Sched. 14(6) (2011), pp. 617–637.

[27] M. Kalender, A. Kheiri, E. Özcan, and E.K. Burke, *A greedy gradient-simulated annealing selection hyper-heuristic*, Soft Comput. 17(12) (2013), pp. 2279–2292.

[28] K.D. Kang, I.K. Moon, and H.F. Wang, *A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem*, Appl. Math. Comput. 219 (2012), pp. 1287–1299.

[29] L. Kang and G.M. White, *A logic approach to the resolution of constraints in timetabling*, European J. Oper. Res. 61(3) (1992), pp. 306–317.

[30] K. Karabulut and M.M. Inceoglu, *A hybrid genetic algorithm for packing in 3D with deepest bottom left with fill method*, AVDIS 3261 (2004), pp. 441–450.

[31] A.H. Karami and M. Hasanzadeh, *University course timetabling using a new hybrid genetic algorithm*, Proceedings of the 2nd International eConference on Computer and Knowledge Engineering, 2012, pp. 144–149.

[32] G. Lach and M.E. Lübbecke, *Curriculum based course timetabling: New solutions to Udine benchmark instances*, Ann. Oper. Res. 194(1), (2012) pp. 255–272.

[33] N.L. Lawrie, *An integer linear programming model of a school timetabling problem*, Comput. J. 12(4) (1969), pp. 307–316.

[34] R. Lewis and B. Paechter, *New crossover operators for timetabling with evolutionary algorithms*, Proceedings of the 5th International Conference on Recent Advances in Soft Computing RASC2004, 2004, pp. 189–194.

[35] R. Lewis and B. Paechter, *Application of the grouping genetic algorithm to university course timetabling*, in *Evolutionary Computation in Combinatorial Optimization*, G.R. Raidl and J. Gottlieb, eds., Springer, Berlin, 2005, pp. 144–153.

[36] Y.K. Liu, D.F. Zhang, and F.Y.L. Chin, *A clique-based algorithm for constructing feasible timetables*, Optim. Methods Softw. 26 (2011), pp. 281–294.

[37] Z. Lü and J.K. Hao, *Adaptive tabu search for course timetabling*, European J. Oper. Res. 200(1) (2010), pp. 235–244.

[38] D. Mack and A. Bortfeldt, *A heuristic for solving large bin packing problems in two and three dimensions*, Cent. Eur. J. Oper. Res. 20 (2012), pp. 337–354.

[39] P. McMullan, *An extended implementation of the great deluge algorithm for course timetabling*, in *Computational Science—ICCS 2007*, Y. Shi, G.D. van Albada, J. Dongarra, and P.M.A. Sloot, eds., Springer, Berlin, 2007, pp. 538–545.

[40] S. MirHassani, *A computational approach to enhancing course timetabling with integer programming*, Appl. Math. Comput. 175(1), (2006) pp. 814–822.

[41] I.K. Moon and T.V.L. Nguyen, *Container packing problem with balance constraints*, OR Spectrum 36 (2014), pp. 837–878.

[42] J.M. Mulvey, *A classroom/time assignment model*, European J. Oper. Res. 9(1) (1982) pp. 64–70.

[43] F. Parreño, R. Alvarez-Valdes, J.F. Oliveira, and J.M. Tamarit, *A hybrid GRASP/VND algorithm for two- and three-dimensional bin packing*, Ann. Oper. Res. 179 (2010), pp. 203–220.

[44] O. Rossi-Doria, M. Sampels, M. Birattari, M. Chiarandini, M. Dorigo, L.M. Gambardella, and B. Paechter, *A comparison of the performance of different metaheuristics on the timetabling problem*, in *Practice and Theory of Automated Timetabling IV*, E. Burke and P. De Causmaecker, eds., Springer, Berlin, 2003, pp. 329–351.

[45] K. Schimmelpfeng and S. Helber, *Application of a real-world university-course timetabling model solved by integer programming*, OR Spectrum 29(4) (2007), pp. 783–803.

[46] K. Socha, J. Knowles, and M. Sampels, *A max-min ant system for the university course timetabling problem*, in *Ant Algorithms*, M. Dorigo, G. Di Caro, and M. Sampels, eds., Springer, Berlin, 2002, pp. 1–13.

[47] J.A. Soria-Alcaraz, G. Ochoa, J. Swan, M. Carpio, H. Puga, and E.K. Burke, *Effective learning hyper-heuristics for the course timetabling problem*, European J. Oper. Res. 238(1), (2014), pp. 77–86.

[48] A. Tripathy, *School timetabling—a case in large binary integer linear programming*, Manage. Sci. 30(12) (1984), pp. 1473–1489.

[49] H. Turabieh, S. Abdullah, and B. Mccollum, *Electromagnetism-like mechanism with force decay rate great deluge for the course timetabling problem*, in *Rough Sets and Knowledge Technology*, P. Wen, Y. Li, L. Polkowski, Y. Yao, S. Tsumoto, and G. Wang, eds., Springer, Berlin, 2009, pp. 497–504.

[50] J. Wahid and N.M. Hussin, *Harmony Search Algorithm for Curriculum-Based Course Timetabling Problem*, arXiv preprint arXiv:1401.5156, 2014.

[51] R. Weare, E. Burke, and D. Elliman, *A hybrid genetic algorithm for highly constrained timetabling problems*, The Proceedings of the Sixth International Conference on Genetic Algorithms, ed. LJ Eshelman, 1995.

[52] D.F. Zhang, Y.K. Liu, R. M'Hallah, and S.C.H. Leung, *A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems*, European J. Oper. Res. 203 (2010), pp. 550–558.

[53] W.B. Zhu and A. Lim, *A new iterative-doubling Greedy—Lookahead algorithm for the single container loading problems*, European J. Oper. Res. 222 (2012), pp. 408–417.