Eskişehir Osmangazi Üniversitesi Mühendislik ve Mimarlık Fakültesi Dergisi 27(2), 67 - 76, 2019

Journal of Engineering and Architecture Faculty of Eskisehir Osmangazi University 27(2), 67 - 76, 2019

A NEW MATHEMATICAL MODEL AND RANDOM KEY BASED METAHEURISTIC SOLUTION APPROACH FOR COURSE-ROOM-TIME ASSIGNMENT PROBLEM

Zehra KAMIŞLI ÖZTÜRK[1*], Mujgan SAĞIR[2]

[1]Eskisehir Technical University, Faculty of Engineering, Department of Industrial Engineering, Eskisehir, Turkey. ORCID No : http://orcid.org/0000-0003-3156-6464

[2]Eskisehir Osmangazi University, Engineering and Architecture Faculty, Department of Industrial Engineering Eskisehir – Turkey, ORCID No : http://orcid.org/0000-0003-2781-658X

DOI : http://dx.doi.org/10.31796/ogummf.549986

Keywords

Course-room-time assignment problem, Mathematical modelling, Random key based genetic algorithms.

Abstract

This study presents a newly developed mixed-integer mathematical model for university course-room-time assignment problem. Optimal results with no soft constraint violations are obtained for some type of problem instances. As problem complexity increases it becomes more difficult to find feasible solution for this problem in a reasonable time. Therefore, a heuristic approach is often needed for such problems. In this study, a random key based genetic algorithm (RKGA) is developed. RKGA encoding is used in order to encode the chromosomes with a length of just the number of courses and not to use problem specific genetic operators and/or repair mechanisms. Well-known problem instances from the literature are selected to evaluate the outcome. The performance of RKGA is competitive to that of other algorithms especially for big size problems.

DERS-DERSLİK-ZAMAN DİLİMİ ATAMA PROBLEMİ İÇİN YENİ BİR MATEMATİKSEL MODEL VE RASSAL ANAHTAR TEMELLİ METASEZGİSEL ÇÖZÜM YAKLAŞIMI

Anahtar Kelimeler

Ders-derslik-zaman dilimi atama problemi, Matematiksel modelleme, Rassal anahtar temelli genetik algoritma.

Öz

Bu çalışmada üniversite ders-derslik-zaman dilimi atama problemi için yeni bir karma tam sayılı matematiksel model önerilmiştir. Geliştirilen karma tam sayılı matematiksel model ile literatürde yer alan test problemleri çözdürülmüş ve bir kısmı için tüm esnek kısıtlar sağlanarak en iyi çözüm elde edilmiştir. Problem karmaşıklığı arttıkça makul sürelerde uygun çözüm bulmak zorlaştığından, bu tür problemlerin çözümü için sezgisel bir yaklaşıma ihtiyaç duyulmaktadır. Çalışmada, rassal anahtar temelli bir genetik algoritma (RKGA) geliştirilmiştir. Probleme özgü özel genetik operatörler ve/veya onarma mekanizmaları kullanmamak için sadece ders sayısı uzunluğundaki kromozomları kodlamak için RKGA kodlaması kullanılmıştır. Çıktıların değerlendirilmesi için literatürde iyi bilinen test problemleri seçilmiştir. Özellikle büyük boyutlu problemlerde RKGA'nın performansının diğer algoritmalar ile rekabet edebilir düzeyde olduğu görülmüştür.

Araştırma Makalesi		Research Article	
Başvuru Tarihi	: 05.04.2019	Submission Date	: 05.04.2019
Kabul Tarihi	: 13.06.2019	Accepted Date	: 13.06.2019

1. Introduction

Timetabling problems are a kind of scheduling problems deal with allocation of given resources to objects being placed in time, in such a way as to satisfy as nearly as possible a set of desirable objectives, subject to constraints. Educational timetabling problems, in this concept, are classified into two main groups: course and examination scheduling. In general, course scheduling is defined as assigning courses to rooms and timeslots in such a way that there are no conflicts or clashes by satisfying objectives and the examination timetabling basically involves allocating a set of examinations to a set of rooms and time periods. The basic difference is that in course timetabling there cannot be more than one course per room in contrary of exam one.

Since 1960's, when the timetabling problems have

* Corresponding author; e-mail: zkamisli@eskisehir.edu.tr

appeared as of interest for many researches, the educational timetabling systems are required to be more flexible by adding elective courses and considering personal preferences. Also, real-world timetabling systems have to cope with much more challenging requirements, such as "students should not have gaps in their individual daily timetables", which often make the problem over-constrained (Burke, Marecek, Parkes and Rudová, 2007a). In general, the construction of a schedule is an optimization problem of arranging time, space, and (often limited) resources simultaneously. However, as Gunawan, Ng and Poh (2007) mentioned, mathematical programming models are not an effective way for finding the existence of an optimal solution, especially for large-scale timetabling problems. Thus, the design of heuristic approaches is proposed.

The outline of the paper is as follows. The university course-room-time assignment problem and the proposed mathematical model are introduced in Section 2. The proposed random key based genetic algorithm (RKGA) is also given in Section 2. The results are given in Section 3. Finally, Section 4 summarizes the conclusions.

2. Material and Method

To compare the performance of our proposed methodology with the current methods, we consider the course-room-time assignment problem studied within the Metaheuristics Network and by many researches like Socha, Knowles and Samples(2003), Acha and Nieuwenhuis (2014), Abuhamdah, Ayob, Kendall and Sabar (2014). The problem instances defined by Ben Paechter (http://www.metaheuristics.net/index.php%3Fmain=4&sub=44.html). These problem instances are classified in three groups as *small*, *medium* and *large*.

Problem definition is given as follows: There is a set of courses to be scheduled in 45 timeslots as nine for each of five days, a set of rooms for courses, a set of students attending the courses and a set of features required by these courses and features included in rooms. Each course spans an hour, each student attends a number of courses and each room has a capacity. Table 1 presents the main characteristics of test problems defined by Paechter. In this study, an "event" refers to a "course".

Table 1

Characteristics of the Test Problems

	Small	Medium	Large
#courses	100	400	400
#rooms	5	10	10
#features	5	5	10
#students	80	200	400

Scheduling allocates resources to activities over time based on hard and/or soft constraints. The hard constraints are listed as follows:

(1) No student should be assigned to more than one event at a timeslot.

(2) The room assigned to an event should have sufficient capacity and all the features required by that event.

(3) At most one event can be scheduled in one room at a timeslot.

Besides, to improve the solution quality and the overall performance of the educational system, three soft constraints are imposed as listed below. These constraints are preferred to be satisfied as much as possible.

(1) A student is not preferred to have more than two consecutive classes on a day.

(2) A student is not preferred to have only a single class on a day.

(3) A student is not preferred to have a class in the last time slot of a day.

The quality of timetable is measured by penalizing each violation of the soft constraints where each violation will be penalized by '1' for each student who involves in this situation.

2.1. Mathematical Model

A mixed-integer mathematical model is developed for the defined problem. Model parameters, decision variables, constraints and the objective function are given below. The time slots are numbered as seen in Table 2.

Table 2

Timeslots of Days

Hours	Monday	Tuesday	Wednesday	Thursday	Friday
1	1	10	19	28	37
2	2	11	20	29	38
3	3	12	21	30	39
4	4	13	22	31	40
5	5	14	23	32	41
6	6	15	24	33	42
7	7	16	25	34	43
8	8	17	26	35	44
9	9	18	27	36	45

Sets and indices

J = $\{j \mid j=1,\dots, n\}$ for courses
I = $\{i \mid i=1,\dots, m\}$ for students
K = $\{k \mid k = 1,\dots, h\}$ for rooms
L = $\{l \mid l = 1,\dots , o\}$ for features
T = $\{t \mid t=1,\dots, 45 \}$ for timeslots
D = $\{d \mid d=1,\dots, 5 \}$ for days

Tlast = {*t* | *t* = 9, 18, 27, 36, 45} Last time slot of a day,

Tlast ∈ T

Parameters
r_k :the capacity of the k^{th} room
OD: the 0-1student-course matrix
SN: the 0-1room-feature matrix
DN: the 0-1course-feature matrix
M : a positive big number

If course *j* is attended by student *i*, corresponding entry of OD matrix, OD_{ij} is equal to 1. If room k has feature *l*, corresponding entry of SN matrix, SN_{kl} is equal to 1, and similarly, if course j requires feature l, corresponding entry of DN matrix, DN_{jl} is equal to 1.

Decision variables

$$y_{jk} = \begin{cases} 1, & if\ course\ j\ is\ assigned\ to\ room\ k \\ 0, & o.w. \end{cases}$$

$$z_{jt} = \begin{cases} 1, & if\ course\ j\ is\ assigned\ to\ time\ t \\ 0, & o.w. \end{cases}$$

$$x_{jkt} = \begin{cases} 1, & if\ course\ j\ is\ assigned\ to\ room\ k\ on\ time\ t \\ 0, & o.w. \end{cases}$$

$$rr_{it} = \begin{cases} 1, & if\ student\ i\ has\ course\ on\ time\ t \\ 0, & o.w. \end{cases}$$

$$y3_{id} = \begin{cases} 1, & if\ student\ i\ has\ only\ one\ scheduled\ course\ on\ day\ d \\ 0, & o.w. \end{cases}$$

$ya2_{id} \in \{0,1\}$
$y1_{id} \in \{0,1\}$
$y2_{id} \in \{0,1\}$
$y3_{id} \in \{0,1\}$
$ya2_{id}, y1_{id}, y2_{id}$ and $y3_{id}$ are the pseudo variables used to consider soft constraint violations.

$d1, d2$ and $d3$ are the variables represent the soft constraint violations.

$d1$: The total number of occurrences that students have a course in the last time slot of days

$d2$: The total number of occurrences that students have more than two consecutive courses for all days

$d3$: The total number of occurrences that students have only a single course on a day

In the light of the above definitions, the mathematical model of the course-room-time assignment problem is given as following:

$$min\ Z = d1 + d2 + d3 \tag{1}$$

subject to

$$y_{jk} DN_{jl} \le SN_{kl} \qquad \forall (j,k,l) \tag{2}$$

$$\sum_t x_{jkt} \ge y_{jk} \qquad \forall (j,k) \tag{3}$$

$$\sum_t x_{jkt} \le M * y_{jk} \qquad \forall (j,k) \tag{4}$$

$$\sum_k \sum_t x_{jkt} = 1 \qquad \forall j \tag{5}$$

$$\sum_j x_{jkt} \le 1 \qquad \forall (k,t) \tag{6}$$

$$\sum_i OD_{ij} y_{jk} \le r_k \qquad \forall (j,k) \tag{7}$$

$$\sum_j OD_{ij} z_{jt} \le 1 \qquad \forall (i,t) \tag{8}$$

$$\sum_k x_{jkt} \ge z_{jt} \qquad \forall (j,t) \tag{9}$$

$$\sum_k x_{jkt} \le M * z_{jt} \qquad \forall (j,t) \tag{10}$$

$$rr_{it} = \sum_j OD_{ij}\, z_{jt} \qquad \forall\ (i,t) \tag{11}$$

$$\sum_i \sum_{t \epsilon Tlast} rr_{it} = d1 \tag{12}$$

$$\sum_{(d-1)*9+1 \le t \le (d-1)*9+3} rr_{it} \ge 3 * ya2_{id} \qquad \forall (i,d) \tag{13}$$

$$\sum_{(d-1)*9+2 \le t \le (d-1)*9+4} rr_{it} \ge 3 * ya2_{id} \qquad \forall (i,d) \tag{14}$$

$$\sum_{(d-1)*9+3 \le t \le (d-1)*9+5} rr_{it} \ge 3 * ya2_{id} \qquad \forall (i,d) \tag{15}$$

$$\sum_{(d-1)*9+4 \le t \le (d-1)*9+6} rr_{it} \ge 3 * ya2_{id} \qquad \forall (i,d) \tag{16}$$

$$\sum_{(d-1)*9+5 \le t \le (d-1)*9+7} rr_{it} \ge 3 * ya2_{id} \qquad \forall (i,d) \tag{17}$$

$$\sum_{(d-1)*9+6 \le t \le (d-1)*9+8} rr_{it} \ge 3 * ya2_{id} \qquad \forall (i,d) \tag{18}$$

$$\sum_{(d-1)*9+7 \le t \le (d-1)*9+9} rr_{it} \ge 3 * ya2_{id} \qquad \forall (i,d) \tag{19}$$

$$\sum_{(d-1)*9+1 \le t \le (d-1)*9+3} rr_{it} \le 3 * ya2_{id} + 2 * (1 - ya2_{id}) \qquad \forall (i,d) \tag{20}$$

$$\sum_{(d-1)*9+2 \le t \le (d-1)*9+4} rr_{it} \le 3 * ya2_{id} + 2 * (1 - ya2_{id}) \qquad \forall (i,d) \tag{21}$$

$$\sum_{(d-1)*9+3 \le t \le (d-1)*9+5} rr_{it} \le 3 * ya2_{id} + 2 * (1 - ya2_{id}) \qquad \forall (i,d) \tag{22}$$

$$\sum_{(d-1)*9+4 \le t \le (d-1)*9+6} rr_{it} \le 3 * ya2_{id} + 2 * (1 - ya2_{id}) \qquad \forall (i,d) \tag{23}$$

$$\sum_{(d-1)*9+5 \le t \le (d-1)*9+7} rr_{it} \le 3 * ya2_{id} + 2 * (1 - ya2_{id}) \quad \forall(i,d) \quad (24)$$

$$\sum_{(d-1)*9+6 \le t \le (d-1)*9+8} rr_{it} \le 3 * ya2_{id} + 2 * (1 - ya2_{id}) \quad \forall(i,d) \quad (25)$$

$$\sum_{(d-1)*9+7 \le t \le (d-1)*9+9} rr_{it} \le 3 * ya2_{id} + 2 * (1 - ya2_{id}) \quad \forall(i,d) \quad (26)$$

$$\sum_{i} \sum_{d} ya2_{id} = d2 \quad (27)$$

$$\sum_{(d-1)*9 \le t \le d*9} rr_{it} \ge 2 * y2_{id} + y3_{id} \quad \forall(i,d) \quad (28)$$

$$\sum_{(d-1)*9 \le t \le d*9} rr_{it} \le y3_{id} + 9 * y2_{id} \quad \forall(i,d) \quad (29)$$

$$y1_{id} + y2_{id} + y3_{id} = 1, \quad \forall(i,d) \quad (30)$$

$$\sum_{i} \sum_{d} y3_{id} = d3 \quad (31)$$

$$x_j \in \{0,1\} \quad \forall(j,k,t) \quad (32)$$

$$y_{jk} \in \{0,1\} \quad \forall(j,k) \quad (33)$$

$$z_{jt} \in \{0,1\} \quad \forall(j,t) \quad (34)$$

$$rr_{it} \in \{0,1\} \quad \forall(i,t) \quad (35)$$

$$y1_{id} \in \{0,1\} \quad \forall(i,d) \quad (36)$$

$$y2_{id} \in \{0,1\} \quad \forall(i,d) \quad (37)$$

$$y3_{id} \in \{0,1\} \quad \forall(i,d) \quad (38)$$

$$ya2_{id} \in \{0,1\} \quad \forall(i,d) \quad (39)$$

$$d1, d2, d3 \ge 0 \quad (40)$$

With constraint (2) it's provided that the courses are assigned only to the rooms that include required features. If any course-room pair (j, k) is assigned to some time slot, constraint set (4) forces y_{jk} to be equal to 1. If this pair is not assigned to any time slot, then the left-hand sides of the corresponding two inequalities in constraint sets (3) and (4) are zero, and hence, the constraint set (3) forces y_{jk} to be equal to 0. A course is assigned to only one room and one timeslot by constraint (5). It's provided that only one course can be assigned to each room at any timeslot by constraint (6). With the constraint (7) the courses are assigned to suitable rooms by means of the capacities. And finally, constraint (8) ensures that a student does not take more than one course in a timeslot.

If any course-time slot pair (j, t) is assigned to a room the constraint set (10) forces z_{jt} to be equal to 1. If this pair is not assigned to any room, then the left-hand sides of the corresponding two inequalities in the constraint sets (9) and (8) are zero, and hence, the constraint set (9) forces z_{jt} to be equal to 0.

If any course-time slot pair (j, t) is assigned to a room the constraint set (10) forces z_{jt} to be equal to 1. If this pair is not assigned to any room, then the left-hand sides of the corresponding two inequalities in the constraint sets (9) and (8) are zero, and hence, the constraint set (9) forces z_{jt} to be equal to 0.

As a soft constraint, courses are not preferred to be assigned to the last time slot of a day. The constraint (11) holds the value of rr_{it}, that student i has a course at time t or hasn't. Constraint (12) calculates d1 as the total number of courses assigned to last time slots of all days for all students by using rr_{it} .

As a second soft constraint, students are not preferred to have more than two consecutive courses in a day. All three consecutive timeslots of a day are taken into consideration to check whether if a student has more than two consecutive courses in a day or not. If a student has three consecutive courses in a day the variable ya2id takes the value 1. There are 9 timeslots daily so, seven different consecutiveness situations are required to be checked for each day. The constraints (13)-(26) represent the consecutiveness. For constraint (13), consider the first day (d=1). If d=1, then 1≤t≤3 means that three consecutive timeslots begin at time 1 and ends at time 3. If d=2, then 10≤t≤12 means that three consecutive timeslots begin at time 10 and ends at time 12, and so on. The other constraints (14)-(19) check the consecutiveness beginning at time slot 2, 3, 4, 5, 6, 7 and 9 for each day. The constraint (27) calculates d2 as the total number of situations that more than two consecutive courses assigned to all students in all days.

Students are not preferred to have only one scheduled course on a day and this requirement is hold by the last soft constraint. A student may not have any course in a day either. In this case, the variable $y1_{id}$ takes the value 1, and the variables $y2_{id}$ and $y3_{id}$ take the value 0. If a student has a single course on day d, the variable $y3_{id}$ takes the value 1, the variables $y2_{id}$ and $y1_{id}$ take the value 0. Both situations are provided by the constraints (28)-(30). The constraint (31) calculates $d3$ as the total number of occurrences that having a single course on a day for all students.

The objective of the course-room-time assignment problem is defined as minimizing the soft constraint violations. By definition, as explained above, $d1$, $d2$ and $d3$ are the number of soft constraint violations which then used in the objective function as minimizing their weighted sum with equal weights.

2.2. Problem Complexity and the Need for a Heuristic Approach

The optimum solutions that satisfy all soft constraints are obtained by using GAMS with CPLEX solver for all small instances. Small type test problems are run in average 3.5 hours by an 8-core MacPro computer with 2.13 GHz and 6 GB RAM.

On the other hand, we were able to solve this problem for 100 courses in three and a half hours. By being encouraged by this improvement, Medium type problems which have 400 courses are tried to be solved. However, no feasible solution is found in 96 hours and the running process is then terminated. We present some sets of solution performances for Small instances in Table 3. Burke, Kendall and Soubeiga (2003) and Socha et al., (2003) solved the problem. The numbers in columns refer the number of soft constraint violations. It's seen that our approach is able to solve all Small types of the problem without any soft constraint violation.

Table 3

Some Solution Performances for Small Type Instances

instances	HH [4]	RRLS [5]	ANT [5]	proposed mathematical model
Small01	1	8	1	0
Small02	2	11	3	0
Small03	0	8	1	0
Small04	1	7	1	0
Small05	0	5	0	0

In his literature survey, Schaerf (1999) explains that the optimum solution for course-room-time assignment problem is obtained only for small instances up to ten courses. It's seen that since his statement, the good quality solutions of some are even optimal is obtained for the defined problem. For instance, Daskalaki, Birbas and Housos (2004) proposed an integer programming formulation. In order to evaluate the proposed "one-objective IP model", three problems of different size were solved. The number of the courses varied from 25 to 92 in addition to the lab courses that varied from 8 to 27, totaling the requirements for teaching periods from 139 to 326. We should note that these teaching periods are scheduled within the 70 available time periods during each week. The mode suggested IP formulation carried 7,543–17,159 equations and 4,100–19,295 binary variables, while the non-zeros of the IP model varied from 35,685 to 92,358. Similarly, Schimmelpfeng and Helber (2007) also were able to solve the course scheduling problem at School of Economics and Management at Hannover University, Germany. The decision problem is to assign teaching groups to time slots and rooms so that several soft and hard constraints are met.

By defining i as student, j as course, k as room, l as feature and t as timeslot; the proposed model has ($3jk + 2jt + jkl + j + kt + 2it + 17id + 2$) number of constraints and ($jkt + jk + jt + it + 4id + 3$) number of variables.

For the illustrative example Small01 (j=100, i=80, k=5, l=5, t=45, d=5), there are 27322 constraints and 32703 variables. For *Medium* type problems (j=400, i=200, k=10, l=5, t=45, d=5), these values will be 67852 and 215003 respectively. Even for the smallest model with 100 courses, the problem size is quite large. As the variable and the number of constraints increase, the search space and the complexity of the problem will substantially increase.

There are various techniques developed to solve the problem. While smaller instances might be solved by exact algorithms, most real-world problems are large dimensional problems, so there is a need for heuristic methods to obtain near-optimal solutions in reasonable time. The most used ones are metaheuristics like tabu search (Valdes, Crespo and Tamarit, 2002; Yuan and Lan, 2016), simulated annealing (Abramson, 1991; Thompson and Dowsland, 1998; Bellio, Ceschia, Di Gaspero, Schaerf and Urli, 2016; Goh, Kendall and Sabar 2018), genetic and evolutionary algorithms (Beligiannis, Moschopoulosa, Kaperonisa and Likothanassisa, 2008; Susan and Bhutani, 2018; Matias, Fajardo and Medina, 2018), neural networks (Kovačič, 1993), ant colonies (Socha et al., 2003), bee colony algorithm (Bolaji, Kahader and Betar, 2014), particle swarm optimization (Chen and Shih, 2013;Imran Hossain, Akhand, Shuvo, Siddique and Adeli,, 2019), artificial immune algorithm (Yazdani, Naderi and Zeinali, 2017) and hyperheuristics (Burke, McCollum, Meisels, Petrovic, and Qu, 2007b). Besides, there are some studies dealing with the analysis and design of interactive decision support system for timetable management (Piechowiak and Kolski, 2004; Kamisli Ozturk, Ozturk and Sagir, 2010).

In the following section, a random key based genetic algorithm for the solution of related problem is presented.

2.3. Random Key Based Genetic Algorithm (RKGA) Approach for Course-Room-Time Assignment

Genetic algorithm search methods are rooted in the mechanisms of evolution and natural genetics. They generate a sequence of populations by using a selection mechanism, and use crossover and mutation as search mechanisms. In the literature, Holland's genetic algorithm is commonly called as the Simple Genetic Algorithm (SGA). Essential to the SGA's working is a population of binary strings. Each string of 0s and 1s is the encoded version of a solution to the optimization problem (Srinivas and Patnaik, 1994).

The encoding strategy is different for different optimization problems, and a given problem may have more than one workable encoding strategy (Snyder and Daskin, 2006). A good encoding method can make finding good solutions relatively efficient. Conversely, a poor chromosome encoding method can make finding good solutions nearly impossible (Michalewicz, 1996; Eklund, 2006). The coding procedure for the educational timetabling problems solved by GA in the literature is mostly based on matrix and vector. When the binary coding is used, the chromosome length is the "number of course times the number of resources". In another issue encountered is the probability of crossover and mutation operators to generate infeasible solutions. These infeasible solutions may be the result of using an inappropriate chromosome representation and traditional genetic operators. Therefore, special crossover and mutation operators use different representation and genetic repair mechanisms are required.

These stressed drawbacks can be overcome by transferring the concept of random keys. The random key representation for representing permutations was first presented by Bean (1994). RKGA encoding is used to encode the chromosomes of length n. This encoding has the important property of never producing infeasible solutions to permutation problems either in the initial population or through any crossover or mutation operation. As a result, no repair mechanism or problem-specific operators are needed. Each gene represents a particular sampled value, and the alleles are initially of random numbers. The sorted order of the alleles determines to which position (either a particular position in the hole or set aside) the sampled value is assigned (Eklund, 2006).

RKGAs are used for various problems like single and multiple machine scheduling, vehicle routing, resource allocation, quadratic assignment, traveling salesman, facility layout and multi-robot welding task sequencing problems.

The following section introduces a newly developed RKGA for course-room-time assignment problem, considered in this paper.

As Bean proposed, Figure 1 depicts the entire general transition of a RKGA.

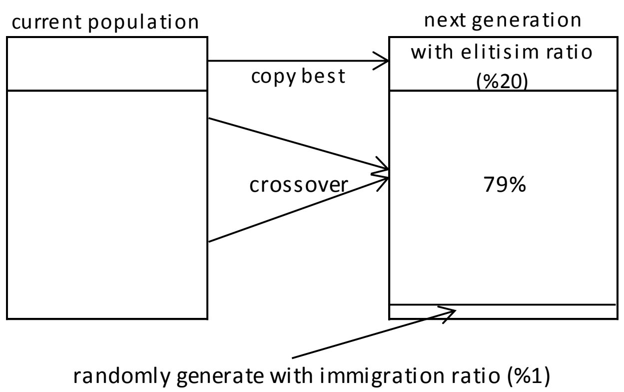

Figure 1. General Transition (Bean, 1994)

In the first step of the algorithm, initial population is generated by using random keys. Then all chromosomes are decoded and evaluated. In order to construct next generation, elitism, crossover and immigration operators are applied to the current population according to predetermined ratios. The algorithm continues up to a point of a specified number of iterations. The basic steps are explained in detail as follows:

2.3.1 Initialization

Since there are 45 available time slots for each room, we can't excess this number for course-room assignments. In the room assignment procedure, each gene is used for assignments one by one. For the last assignment we assign the room enforcedly for the last course. If this course has a special requirement that room doesn't satisfy, an infeasible solution is obtained. In order to avoid such solutions we want to assign courses to rooms by considering this rule. As the initialization step, a feasible course-room assignment is obtained. The assignment also provides assigning an equal number of courses to each room as much as possible by considering course clashes and requirements.

2.3.2 Chromosome structure and decoding

In the RKGA proposed here, each chromosome consists of genes up to the total number of courses to be scheduled. Figure 2 gives a chromosome for a hundred courses. Each gene on the chromosome is encoded by a four digit random number. In order to decode a chromosome to the solution of the problem we suggest a decoding procedure as follows:

First of all, we divide each random number into two parts as rs_1 and rs_2. The first two digits (rs_1) are related with room and the last two (rs_2) are for the time assignments.

course no:	*1*	*2*	*3*	*4*	*...*	*100*
room & time:	1335	2357	1190	7809	...	9813

Figure 2. A Chromosome for a Hundred Number of Courses

Two matrices are organized: a matrix of courses, CT, can be assigned to each timeslot, and a matrix of courses, CR, can be assigned to each room.

The courses are assigned to rooms according to the second matrix. In this process, the courses are sorted by their total number of available rooms that each satisfies the feature requirements of courses. Hence an assignment probability o_{jk} which is calculated based on the number of courses assigned by that specified phase of the current iteration is used.

By using the room matrix that the course *j* can be assigned to (cr_{jk}), and total number of courses assigned to room *k* by that time (nd_k), the assignment probability can be calculated by equation (41). If rs_1 is smaller than the $o_{jk} * 100$, the course *j* is assigned to room *k*. Otherwise, next available room is checked. Until all the courses are assigned, the procedure continues, similarly.

$$total = \sum_j \sum_k cr_{jk} * (45 - nd_k)$$

$$o_{jk} = \frac{cr_{jk} * (45 - nd_k)}{total} \quad (41)$$

Followed by course-room assignments, the course-room pairs are assigned to timeslots by considering soft and hard constraints. The last time slots of each day are considered as lay over and the courses have common students are not allowed to be assigned to the same timeslots.

Let us give some more details related to this procedure. The course *j* as the first course of the randomly ordered course set is selected. Its related courses are defined as next. Related courses are the ones have common students. If the course *j* doesn't have any such course, its assignable time slot is calculated by rs_2. Then course *j* is assigned to this defined time period. Otherwise, related courses are being prevented from being assigned to this specific time period.

Consider the chromosome given in Figure 3. In the first gene (1335), the last two digit of the random key "35" is used for the time assignment. According to the assignment procedure, we previously determine that the first course can be assigned to three time slots as 13, 19 and 44. By using the random key "35", one is selected according to the formula (42).

$$\left\lfloor \frac{(last\ two\ digit\ of\ random\ key) * (number\ of\ available\ time\ slot}{100} \right\rfloor + 1 \quad (42)$$

As given in the Figure 3, the first course is assigned to the timeslot 19 which is in the second order.

course no:	1	2	3	4	...	100
room & time:	1335	2357	1190	7809	...	9813

	order
13	1
$\lfloor (35*3)/100 \rfloor + 1 = 2 \rightarrow$ 19	2
44	3

Figure 3. Timeslot Assignment Using Random Keys

2.3.3 Evaluation of the fitness value

The evolution process is conducted to accomplish the objectives (minimization of the three soft constraint violations) of the problem. The fitness value for the solutions is calculated as the sum of $f1$, $f2$ and $f3$, where the $f1$, $f2$ and $f3$ represent the soft constraint violations, described in the previous section, respectively.

2.3.4 Crossover

Instead of traditional one/multi point crossover operator, parametric crossover is used. Parametric crossover is applied according to the RKGA's parameter of head probability and a new chromosome is generated from randomly selected two chromosomes. Then, for each gene of the new chromosome, random numbers are generated. If the random number is smaller than the head probability, the gene of the new chromosome is copied from the first chromosome otherwise from the second one. A sample of parametric crossover where the head probability is 0.6 is given in Figure 4. For instance, the first random number is smaller than the head probability (0.17<0.6), so the first gene of the new chromosome is the first gene of the chromosome 1 as 1214, and so on.

Genes	*1*	*2*	*3*	*...*	*N*
Chromosome 1	1214	2241	6173	...	1829
Chromosome 2	1132	4342	3324	...	4453
Random number	.17	.77	.89	...	.54
New chromosome	1214	4342	3324	...	1829

Figure 4. Illustration of the Parametric Crossover Operator for Two Chromosomes

3. Computational Results

Small, medium and large problem instances are solved on an 8-core MacPro computer with 2.13 GHz and 6 GB RAM. The results obtained by some algorithms in the literature and RGKA are given and the best ones are highlighted in Table 4. These methods are tabu search based hyper heuristic (HH)4, RRLS and ANT5, fuzzy multiple heuristic ordering (FMHO) (Asmuni, Burke and Garibaldi, 2005), graph-based hyper heuristic (GBHH) (Burke, McCollum, Meisels, Petrovic and Qu, 2007b), a die-hard co-operative ant behavior approach (DCABA) (Ejaz and Javed, 2007), variable neighborhood search (VNS) (Abdullah, Burke and Collum, 2005) and particle collision algorithm (PCA) (Abuhamdah and Ayob, 2005).

In the literature, the comparisons of the results are generally given as number of soft constraint violations. Hence, Table 4 gives the algorithm performances in terms of soft constraint violations except HH algorithm. In HH column not only the soft constraint violations but also the proportions of feasible solutions in 5 runs and best hard constraint violations in all runs are provided. For example, for Small01 instance the proportion of feasible solution in 5 runs is 1, average soft constraint violation is 2.2, and best hard constraint violation is 1. Besides, it's seen that some algorithms are not able to provide feasible solutions for some problem instances.

For example, RRLS is one of these algorithms for Medium05 and Large problems and also VNS for Large.

DCABA obtains feasible results competitor to others for some instances (Small05). VNS approach obtains the best results for small type instances. However, it couldn't find a solution for the Large instance and the results for Medium type instances are not as good as the others.

With the parameters of 0.1 elitism ratio, 0.1 immigration ratio, 0.8 head probability and the population size of 20, the best results in 20 runs for RKGA are given in Table 4. The results are quite good for some instances when compared to other algorithms.

RKGA gives better results than RRLS, GBHH and VNS for Medium03 instance. For Medium05 instance, RKGA is better than RRLS. And for Large, RGKA is better than FMHO, RRLS, VNS and HH. In the HH algorithm hard constraint violations can be occurred so it can produce infeasible solutions. Similarly, the RRLS and VNS algorithms are able to produce infeasible solutions. RKGA guarantees the feasible solutions in every run for every instance and produce good-quality solutions by the property of the random keys. Also, the initial population is always feasible and by using the immigration operator, the feasible solutions are not destroyed. Among all methods, the PCA also emphasize that this algorithm doesn't let hard constraint violations.

Table 4
Comparison Between RKGA and Heuristic Results on Course Timetabling Problem Instances

instance	HH	RRLS	ANT	FMHO	GBHH	DCABA	VNS	PCA	RKGA
Small 01	1 / 2.2 / 1	8	1	10	6	5	0	1	20
Small 02	1 / 3 / 2	11	3	9	7	5	0	1	45
Small 03	1 / 1.4 / 0	8	1	7	3	3	0	1	32
Small 04	1 / 1.8 / 1	7	1	17	3	3	0	1	28
Small 05	1 / 0.2 / 0	5	0	7	4	0	0	0	32
Medium 01	1 / 179 / 146	199	195	243	372	176	338	136	530
Medium 02	1 / 197.6 / 173	202.5	184	325	419	154	326	138	523
Medium 03	1 / 295.4 / 267	77,5% Inf	248	249	359	191	384	165	347
Medium 04	1 / 180 / 169	177.5	164.5	285	348	148	299	143	511
Medium 05	0.8 / 388.5 / 303	100% Inf	219.5	132	171	166	307	135	542
Large	0.2 / 1166 / 1166	100% Inf	851.5	1138	1068	798	100%Inf	789	1132

4. Discussion and Conclusion

This study mainly contributes to the literature by presenting a genetic algorithm which is based on random keys as the first RKGA application in the educational timetabling area.

As seen from the literature, just the heuristic solution approaches are proposed. A mathematical model is developed for defined problem as another contribution to the literature, of this study. The developed mathematical model and the optimal solutions for small type instances are given firstly here.

This model is able to obtain optimal solutions of problems for small instances of 100 courses 5 rooms, 5 features and 80 students.

Random keys are efficient tools for encoding ordering and scheduling problems. Based on this motivation the problem is solved by a newly developed RKGA in this study. An important advantage of RKGA is that there is no need for problem specific genetic operators and/or repair mechanisms. The other and also the most important advantage is that it always satisfies feasible solutions in all iterations. It's promising that RKGA gives better results than some of the algorithms as the problem size grows.

For further research, we plan to extend our solution strategies to a few more directions: Some local search algorithms like 2-opt are going to be integrated to the developed algorithm with the hope of having some improved solutions. In addition, developed mathematical model is going to be revised with the hope of having better quality feasible solutions for medium and large sized problems.

There are some assumptions widely used related to educational timetabling researches. A relaxed model will also be considered with course spans more than an hour and also daily course schedules which have lunch breaks.

Acknowledgment

This study is supported by Eskisehir Osmangazi University Scientific Research Projects Committee (Ogubap-project no. 200815009).

Conflict of Interest

No conflict of interest was declared by the authors.

References

Abdullah, S., Burke, E.K. & McCollum, B. (2005). *An investigation of variable neighborhood search for university course timetabling*, Proceedings of 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2005), New York, 413-427. Retrieved from http://www.schedulingconference.org/previous/publications/displaypub.php?key=2005-413-427-P&filename=mista.bib

Abramson, D. (1991). Constructing school timetables using simulated annealing: sequential and parallel algorithms. *Management Science*, 37(1), 98-113. Retrieved from https://dl.acm.org/citation.cfm?id=2845723

Abuhamdah, A. & Ayob, M. (2005). Experimental result of particle collision algorithm for solving course timetabling problems. *International Journal of Computer Science and Network Security*, 9(9), 134-142.

Abuhamdah, A., Ayob, M., Kendall, G. & Sabar, N. (2014). Population Based Local Search for University Course Timetabling Problems. *Applied Intelligence*, (40), 44–53. doi: https://doi.org/10.1007/s10489-013-0444-6

Achá, R. & Nieuwenhuis, R. (2014). Curriculum-Based Course Timetabling with SAT and Maxsat. *Annals of Operations Research*, 218, 71–91. doi: https://doi.org/10.1007/s10479-012-1081-x

Asmuni, H., Burke, E.K. & Garibaldi, J. (2005). *Fuzzy multiple heuristic ordering for course timetabling*, Proceedings of the 2005 UK Workshop on Computational Intelligence UK IC 2005, London, UK, 302-309. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.6678

Bean, J.C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6, 154-160. doi: https://doi.org/10.1287/ijoc.6.2.154

Beligiannis, G.N., Moschopoulosa, C.N., Kaperonisa, G.P. & Likothanassisa, S. D. (2008). Applying evolutionary computation to the school timetabling problem: the Greek case. *Computers and Operations Research*, 35(4), 1265-1280. doi: https://doi.org/10.1016/j.cor.2006.08.010

Bellio, R., Ceschia, S., Di Gaspero, L. Schaerf, A. & Urli, T. (2016). Feature-based tuning of simulated annealing appliedto the curriculum-based course timetabling problem. *Computers & Operations Research*, 65, 83-92. doi: https://doi.org/10.1016/j.cor.2015.07.002

Bolaji, A.L., Kahader, A.T. & Al-Betar, M.A. (2014). University course timetabling using hybridized artificial bee colony with hill climbing optimizer. *Journal of Computational Science*, 5, 809-818. Retrieved from http://dblp.uni-trier.de/db/journals/jocs/jocs5.html#BolajiKAA14

Burke, E.K., Kendall, G. & Soubeiga, E. (2003). A tabu search hyperheuristic for timetabling and rostering. *Journal of Heuristics*, 9(6), 451-470. doi: https://doi.org/10.1023/B:HEUR.0000012446.94732.b6

Burke, E.K., Marecek, J., Parkes, A.J. & Rudová, H. (2007a). Penalising patterns in timetables: novel integer programming formulations. *Operations Research Proceedings*, 2007, 409-414. doi: https://doi.org/10.1007/978-3-540-77903-2_63

Burke, E.K., McCollum, B., Meisels, A., Petrovic, S. & Qu, R. (2007b). A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, 176, 177-192. doi: https://doi.org/10.1016/j.ejor.2005.08.012

Chen, R. & Shih, H. (2013). Solving university course timetabling problems using constriction particle swarm optimization with local search. *Algorithms*, 6, 227-244. doi: https://doi.org/10.3390/a6020227

Daskalaki, S., Birbas, T. & Housos, E. (2004). An integer programming formulation for a case study in university timetabling. *European Journal of Operational Research*, 153, 117-135. doi: https://doi.org/10.1016/S0377-2217(03)00103-6

Ejaz, N. & Javed, M.Y. (2007). *A hybrid approach for course scheduling inspired by die-hard co-operative ant behavior*, Proceedings of the IEEE International Conference on Automation and Logistics, 3095 – 3100. doi: https://doi.org/10.1109/ICAL.2007.4339114

Eklund, N.H.W. (2006). Using genetic algorithms to estimate confidence intervals for missing spatial data. *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, 36(4), 519-524. doi:https://doi.org/10.1109/TSMCC.2006.875407

Goh S.L., Kendall, G & Ssbar, N.R. (2018). Simulated annealing with improved reheating and learning for the post enrolment course timetabling problem. *Journal of the Operational Research Society*, 70(6), 873-888. doi: https://doi.org/10.1080/01605682.2018.1468862

Gunawan, A., Ng, K.M. & Poh, K.L. (2007). Solving the teacher assignment-course scheduling problem by a hybrid algorithm. *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering*, 1(2), 136-141. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.193.3646&rep=rep1&type=pdf

Imran Hossain, Sk., Akhand, M.A.H., Shuvo, M.I.R., Siddique, N.H. & Adeli, H. (2019). Optimization of University Course Scheduling Problem using Particle Swarm Optimization with Selective Search. *Expert Systems with Applications*, 127, 9-24. doi: https://doi.org/10.1016/j.eswa.2019.02.026

Kamisli Ozturk, Z., Ozturk, G. & Sagir, M. (2010). An automated multi-objective invigilator-exam assignment system. *International Journal of Information Technology & Decision Making*, 9(2), 223-238. doi: https://doi.org/10.1142/S0219622010003798

Kovačič, M. (1993). Timetable construction with markovian neural network. *European Journal of Operational Research*, 69, 92-96. doi: https://doi.org/10.1016/0377-2217(93)90094-4

Matias, J.B., Fajardo, A. & Medina, R. (2018). *A Hybrid Genetic Algorithm for Course Scheduling and Teaching Workload Management,* Proceedings of the IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM). https://ieeexplore.ieee.org/document/8666332

Michalewicz, Z. (1996). *Genetic algorithms + data structures = evolution programs*. London: Springer-Verlag.

Piechowiak, S. & Kolski, C. (2004). Towards a generic object oriented decision support system for university timetabling: an interactive approach. *International Journal of Information Technology & Decision Making*, 3(1), 179-208. doi: https://doi.org/10.1142/S0219622004000982

Schaerf, A. (1999). A survey of automated timetabling. *Artificial Intelligence Review*, 13(2), 87-127. doi: https://doi.org/10.1023/A:1006576209967

Schimmelpfeng, K. & Helber, S. (2007). Application of a real-world university-course timetabling model solved by integer programming. *OR Spectrum*, 29, 783-803. doi: https://doi.org/10.1007/s00291-006-0074-z

Socha, K., Knowles, J. & Samples, M. (2003). A max-min ant system for the university course timetabling problem. *Lecture Notes in Computer Science*, 2463(10), 1-13. doi: https://doi.org/10.1007/3-540-45724-0_1

Srinivas, M. & Patnaik, L.M. (1994). Genetic algorithms: a survey. *Computer*, 27(6), 17-26. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=294849

Snyder, L.V. & Daskin, M.S. (2006). A random-key genetic algorithm for the generalized traveling salesman problem. *European Journal of Operational Research*, 174, 38-953. doi: https://doi.org/10.1016/j.ejor.2004.09.057

Susan, S. & Bhutani, A. (2018). *Data Mining with Association Rules for Scheduling Open Elective Courses Using Optimization Algorithms*, In: Abraham A., Cherukuri A., Melin P., Gandhi N. (eds) Intelligent Systems Design and Applications, ISDA 2018, Advances in Intelligent Systems and Computing, 941. doi: https://doi.org/10.1007/978-3-030-16660-1_75

Thompson, J.M., & Dowsland, K.A. (1998). A robust simulated annealing based examination timetabling system. *Computers & Operations Research*, 25(7/8), 637-648. doi: https://doi.org/10.1016/S0305-0548(97)00101-9

Valdes, R.A., Crespo, E. & Tamarit, J.M. (2002). Design and implementation of a course scheduling system using tabu search. *European Journal of Operational Research*, 137, 512-523. doi: https://doi.org/10.1016/S0377-2217(01)00091-1

Yazdani, M., Naderi, B., & Zeinali, E. (2017). Algorithms for university course scheduling problems. *Tehnicki Vjesnik-Technical Gazette*, 24, 241–247. Retrieved from https://www.semanticscholar.org/paper/ALGORITHMS-FOR-UNIVERSITY-COURSE-SCHEDULING-Yazdani-Naderi/8cc10d0845db327726f5bd8abe1d29dce745c3c7

Yuan, Z. & Lan, Q. (2016). Design of course scheduling system based on Tabu Search. *Electronic Design Engineering*, 22. Retrieved from http://en.cnki.com.cn/Article_en/CJFDTotal-GWDZ201622021.htm