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

Zehra KAMİŞLİ ÖZTÜRK¹*, Mujgan SAĞIR²
¹Eskisehir Technical University, Faculty of Engineering, Department of Industrial Engineering, Eskisehir, Turkey.
²Eskisehir Osmangazi University, Engineering and Architecture Faculty, Department of Industrial Engineering
Eskisehir – Turkey, ORCID No : http://orcid.org/0000-0003-3156-6464
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-ZAMÂN DİLİMİ ATAMA PROBLEMI İÇİN YENİ BİR MATEMATİKSEL MODEL VE RASSAL ANAHTAR TEMELLI METASEZGİSEL ÇÖZÜM YAKLAŞIMI

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

Öz

Araştırma Makalesi
Başvuru Tarihi : 05.04.2019
Kabul Tarihi : 13.06.2019

1. Introduction
Timetabling problems are a kind of scheduling problems 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...
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

<table>
<thead>
<tr>
<th></th>
<th>Small</th>
<th>Medium</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>#courses</td>
<td>100</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>#rooms</td>
<td>5</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>#features</td>
<td>5</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>#students</td>
<td>80</td>
<td>200</td>
<td>400</td>
</tr>
</tbody>
</table>

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

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

Sets and indices

\[ J = \{ j \mid j = 1, \ldots, n \} \text{ for courses} \]
\[ I = \{ i \mid i = 1, \ldots, m \} \text{ for students} \]
\[ K = \{ k \mid k = 1, \ldots, h \} \text{ for rooms} \]
\[ L = \{ l \mid l = 1, \ldots, o \} \text{ for features} \]
\[ T = \{ t \mid t = 1, \ldots, 45 \} \text{ for timeslots} \]
\[ D = \{ d \mid d = 1, \ldots, 5 \} \text{ for days} \]
Eskişehir Osmangazi Üniversitesi Mühendislik ve Mimarlık
Fakültesi Dergisi 27(2), 67 - 76, 2019

\[ T_{last} = \{ t \mid t = 9, 18, 27, 36, 45 \} \] Last time slot of a day,
\[ T_{last} \in T \]

**Parameters**
- \( r_k \): the capacity of the \( k^{th} \) room
- \( OD \): the 0-1-student-course matrix
- \( SN \): the 0-1-room-feature matrix
- \( DN \): the 0-1-course-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**
- \( y_{jk} = \begin{cases} 1, & \text{if course } j \text{ is assigned to room } k \text{ o.w.} \\ 0, & \text{otherwise} \end{cases} \)
- \( z_{jt} = \begin{cases} 1, & \text{if course } j \text{ is assigned to time } t \text{ o.w.} \\ 0, & \text{otherwise} \end{cases} \)
- \( x_{jk} = \begin{cases} 1, & \text{if course } j \text{ is assigned to room } k \text{ on time } t \text{ o.w.} \\ 0, & \text{otherwise} \end{cases} \)
- \( r_{it} = \begin{cases} 1, & \text{if student } i \text{ has course on time } t \text{ o.w.} \\ 0, & \text{otherwise} \end{cases} \)
- \( y_{3id} = \begin{cases} 1, & \text{if student } i \text{ has only one scheduled course on day } d \text{ o.w.} \\ 0, & \text{otherwise} \end{cases} \)

\( \begin{align*}
\forall i, d \sum_r r_{it} & \geq 3 \times y_{2id} \\
\forall i, d \sum_r r_{it} & \geq 3 \times y_{2id} + 2 \times (1 - y_{2id}) \\
\forall i, d \sum_r r_{it} & \geq 3 \times y_{2id} + 2 \times (1 - y_{2id}) \\
\forall i, d \sum_r r_{it} & \geq 3 \times y_{2id} + 2 \times (1 - y_{2id}) \\
\forall i, d \sum_r r_{it} & \geq 3 \times y_{2id} + 2 \times (1 - y_{2id}) \\
\forall i, d \sum_r r_{it} & \geq 3 \times y_{2id} + 2 \times (1 - y_{2id}) \\
\forall i, d \sum_r r_{it} & \geq 3 \times y_{2id} + 2 \times (1 - y_{2id}) \\
\forall i, d \sum_r r_{it} & \geq 3 \times y_{2id} + 2 \times (1 - y_{2id}) \\
\forall i, d \sum_r r_{it} & \geq 3 \times y_{2id} + 2 \times (1 - y_{2id}) \\
\end{align*} \]

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

\[ \begin{align*}
\min Z = & \ d1 + d2 + d3 \\
\text{subject to} & \ y_{jk} \cdot DN_{ij} \leq SN_{kl} \quad \forall (j, k, l) \\
\end{align*} \]
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 \(d2\) 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 \(1\leq t\leq 3\) means that three consecutive timeslots begin at time 1 and end at time 3. If \(d=2\), then \(10\leq t\leq 12\) means that three consecutive timeslots begin at time 10 and end 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 \(d3\) 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-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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Small01</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Small02</td>
<td>2</td>
<td>11</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Small03</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Small04</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Small05</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

In his literature survey, Schaefer (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 + 2lt + 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 \((j=400, i=200, k=10, t=5, t=45, d=5)\), 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 (Abrahamson, 1991; Thompson and Dowsland, 1998; Bellio, Ceschia, Di Gaspero, Schaefer and Url, 2016; Goh, Kendall and Sabar (2018)), genetic and evolutionary algorithms (Beligiannis, Moschopoulosa, Kaperonis and Likothenassisa, 2008; Susan and Bhutani, 2018; Matias, Fajardo and Medina, 2018), neural networks (Kovačić, 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, Shuo, Siddique and Adeli, 2019), artificial immune algorithm (Yazdani, Naderi and Zeinal, 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.

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 exceed 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 rs1 and rs2. The first two digits (rs1) are related with room and the last two (rs2) are for the time assignments.

<table>
<thead>
<tr>
<th>course no:</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>...</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>room &amp; time:</td>
<td>1335</td>
<td>2357</td>
<td>1190</td>
<td>7809</td>
<td>...</td>
<td>9813</td>
</tr>
</tbody>
</table>

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 \( o_j \), 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 \((crj)\), and total number of courses assigned to room \( k \) by that time \((ndk)\), the assignment probability can be calculated by equation (41). If \( rs \), 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.

\[
\text{total} = \sum_j \sum_k cr_{jk} \ast (45 - nd_k) \\
o_{jk} = \frac{cr_{jk} \ast (45 - nd_k)}{\text{total}} \quad (41)
\]

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 \( rs_j \). 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).

\[
\left( \text{last two digit of random key} \right) \ast \left( \text{number of available time slot} \right) + 1 \quad (42)
\]

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

<table>
<thead>
<tr>
<th>course no:</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>...</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>room &amp; time:</td>
<td>1335</td>
<td>2357</td>
<td>1190</td>
<td>7809</td>
<td>...</td>
<td>9813</td>
</tr>
<tr>
<td>order</td>
<td>13</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \frac{(35 \ast 3)}{100} )</td>
<td>9</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( +1=2 )</td>
<td>44</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 \( f_1, f_2 \) and \( f_3 \), where the \( f_1, f_2 \) and \( f_3 \) 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.

![Figure 4](image_url)

**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 Collumn, 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.

<table>
<thead>
<tr>
<th>Instance</th>
<th>HH</th>
<th>RRLS</th>
<th>ANT</th>
<th>FMHO</th>
<th>GBHH</th>
<th>DCABA</th>
<th>VNS</th>
<th>PCA</th>
<th>RKGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small 01</td>
<td>1 2.2 / 1</td>
<td>8 1</td>
<td>10 6</td>
<td>5 0</td>
<td>1 20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small 02</td>
<td>1 / 3 / 2</td>
<td>11 3</td>
<td>9 7</td>
<td>5 0</td>
<td>1 45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small 03</td>
<td>1 / 1.4 / 0</td>
<td>8 1</td>
<td>7 3</td>
<td>3 0</td>
<td>1 32</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small 04</td>
<td>1 / 1.8 / 1</td>
<td>7 1</td>
<td>17 3</td>
<td>3 0</td>
<td>1 28</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small 05</td>
<td>1 / 0.2 / 0</td>
<td>5 0</td>
<td>7 4</td>
<td>0 0</td>
<td>0 32</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium 01</td>
<td>1 / 179 / 146</td>
<td>199 195</td>
<td>243 372</td>
<td>176 338</td>
<td>136 530</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium 02</td>
<td>1 / 197.6 / 173</td>
<td>202.5 184</td>
<td>325 419</td>
<td>154 326</td>
<td>138 523</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium 03</td>
<td>1 / 295.4 / 267</td>
<td>77.5% Inf 248</td>
<td>249 359</td>
<td>191 384</td>
<td>165 347</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium 04</td>
<td>1 / 180 / 169</td>
<td>177.5 164.5</td>
<td>285 348</td>
<td>148 299</td>
<td>143 511</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium 05</td>
<td>0.8 / 388.5 / 303</td>
<td>100% Inf 219.5</td>
<td>132 171</td>
<td>166 307</td>
<td>135 542</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large 0.2 / 1166 / 1166</td>
<td>100% Inf 851.5</td>
<td>1138 1068</td>
<td>798 100% Inf</td>
<td>709 1132</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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


