A local search algorithm based on chromatic classes for university course timetabling problem

Velin Kralev* and Radoslava Kraleva
Faculty of Mathematics and Natural Sciences, South-West University, Blagoevgrad, Bulgaria

Received: 25-August-2016; Revised: 10-November-2016; Accepted: 20-November-2016
©2017 ACCENTS

Abstract
This paper presents a study for a local search algorithm based on chromatic classes for the university course timetabling problem. Several models and approaches to resolving the problem are discussed. The main idea of the approach is through a heuristic algorithm to specify the chromatic classes of a graph in which the events of the timetable correspond to the graph vertices and the set of the edges represents the possible conflicts between events. The chromatic classes should be sorted according to specific sort criteria (a total weight or a total count of events in each class), and finally the local search algorithm starts. The aim of the experiments is to determine the best criterion to sort chromatic classes. The results showed that the algorithm generates better solutions when the chromatic classes are sorted in a total weight criterion.

Keywords
University course timetabling, Chromatic classes, Graph-coloring, Local search.

1. Introduction
The University course timetabling problem (UCTP) is a combinatorial problem. For the first time it was presented in [1]. The UCT problem is NP-hard problem [2], but it has a great practical relevance, for instance [3, 4]. It has been shown experimentally that the heuristic approaches give better results than the other approaches for large input data [5-7].

At present a quick algorithm for solving the problem has not been found yet. Though being accurate, the methods based on the approaches “backtracking” and “brute force” (and their variants with decreasing recursion tree among tested partial and complete solutions) can be used only for very small input. In the general case, if there are n events and k time slots, (n-1)!n2k checks for fixing the events on the timetable must be made [8].

There are many approaches to solve approximately the UCTP, for instance:

Constraint-based approaches: In addition, other methods are used, for example: object-oriented modelling of graphs and trees, “depth first search”, combined methods with genetic algorithms and “backtracking” [9].

Case-based reasoning and knowledge-based approaches: In these approaches additional methods are used as well, for example: an expert system of rules and graphs with vertices and edges having attributes that store data on correlations between events [10-12].

Meta-heuristic and hyper-heuristic approaches: These are: "ant colony optimization” [13]; "simulated annealing” [14]; "tabu search” [15]; "variable neighborhood search” [16] and others. The studies are related to finding the most appropriate method for a particular type of problem and the construction of optimal algorithms in terms of computational complexity and memory usage.

Population-based approaches: These are: genetic and memetic algorithms [17] and their modifications [18]. These algorithms build good solutions for an acceptable time.

Graph-based approaches: In these approaches UCTP is represented as a graph coloring problem [3, 19]. A similar approach related to local search method will be discussed in more details below.

*Author for correspondence
2. On the UCTP model
A description of the UCTP model is described in [20]. The model includes matrices, vectors, parameters and soft constraints presented by weights. The UCT problem is mathematically formulated as a multi-criteria optimization problem, and then, by a weighing method, it is transformed into a problem with an objective function. A memetic algorithm (MA with cubic computational complexity) and a genetic algorithm (GA with quadratic computational complexity), are presented in [21]. These algorithms use the model proposed in [20]. After the experiments, it was found out that MA always generates better solutions than GA.

In [22] another algorithm that uses the evaluation model proposed in [20] is presented. In this approach the events are grouped into groups. The best solution depending on the arrangement of the events is searched. Only a very small part of all possible ways of grouping the events have been tested. The grouping of events in groups with an equal number of events in each one is not applicable to all input data sets. For this reason a universal approach how doing this is presented in [23].

3. A graph coloring based approach
The graph-based approaches allow several unrelated groups of events to be processed independently from each other. These events correspond to the graph chromatic classes. In the scientific literature [24, 19], various heuristic techniques for graph coloring adapted for UCTP are described. Let $G = (V, E)$ be an undirected graph with set $V = \{v_1, v_2, ..., v_n\}$ of vertices and set $E = \{(v_i, v_j)| v_i \in V, v_j \in V\}$ of edges, where the unordered pair $(v_i, v_j)$ is an edge in graph $G$ meaning that vertex $v_i$ is adjacent (connected) with vertex $v_j$. (Figure 1).

The events in the timetable correspond to the graph vertices. The set of the edges represent the possible conflicts between events, i.e. every edge shows that its adjacent vertices (events) may not be carried out at the same time, because they use a common resource (student, lecturer or auditorium).

Figure 2 shows how vertices are colored and arranged in chromatic classes. The events in each column may be carried out at the same time, because they are not in conflict with each other. If all events have the same duration – $t$, then the minimum number of time slots that will be needed to conduct all events will be $t \gamma(G)$, where $\gamma(G)$ is the chromatic number of a graph $G$ (i.e. the minimum number of coolers needed to color graph vertices). It should be pointed out that from each permutation of 1, 2, ..., $\gamma(G)$, i.e. $(\gamma(G))!$ an acceptable timetable can be generated [8]. However, in the real UCTP the durations of the events are different.

The graph (vertex) coloring problem is limited to assigning a color (from a given multitude of colors) of each of the vertices of the graph. This problem belongs to the class of NP-hard problems and for its exact solution, it is necessary to use a brute-force method, such as backtracking [19]. For graphs with a large number of vertices and edges (for example more than 50) the exact methods are not effective (regarding performance). In such cases, the heuristic (or approximate) methods are used. In these methods, if the solution found is not optimal, it is close to the
optimal, but it may be satisfactory enough for the given problem [25].

A heuristic approach for coloring the vertices of the graph is proposed in [26]. This method, in graphs, based on the relationships between events, is used in finding the chromatic classes. Initially, the vertices of the graph depending on their degree are sorted in a descending way. Then, from left to right all nonadjacent vertices are colored with the first color. The colored vertices (and the adjacent to them edges) are removed from the graph, and the remaining vertices depending on their degree are sorted in a descending way again. Again from left to right all nonadjacent vertices are colored with a second color, etc., until all vertices of the graph are colored.

An example of the implementation of the algorithm with 35 events, 48 students, 11 lecturers and 8 auditoriums is presented below. Initially, a matrix of the conflicts among events (with shared students, lecturers, or auditoriums) is created. This matrix has dimension $n \times n$ ($n$ is the number of the events). At the first row and the first column of the matrix, the events that correspond to the lectures are located (Table 1). The matrix is symmetric to its main diagonal, and its elements are non-negative whole numbers. The value of each item shows the "cost" of the conflict between events $n_i$ and $n_j$ depending on the quantity of shared resources that are involved in this conflict [20].

![Figure 3 An undirected graph with 35 vertices](image)

<table>
<thead>
<tr>
<th>Event</th>
<th>$E_1$</th>
<th>$E_2$</th>
<th>$E_3$</th>
<th>$E_4$</th>
<th>$E_5$</th>
<th>$E_6$</th>
<th>$E_7$</th>
<th>$E_8$</th>
<th>$E_9$</th>
<th>$E_{10}$</th>
<th>$\ldots$</th>
<th>$E_{35}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>0</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>420</td>
<td>12</td>
</tr>
<tr>
<td>$e_2$</td>
<td>12</td>
<td>0</td>
<td>144</td>
<td>192</td>
<td>192</td>
<td>144</td>
<td>144</td>
<td>120</td>
<td>120</td>
<td>12</td>
<td>420</td>
<td>12</td>
</tr>
<tr>
<td>$e_3$</td>
<td>12</td>
<td>144</td>
<td>0</td>
<td>192</td>
<td>192</td>
<td>144</td>
<td>144</td>
<td>120</td>
<td>120</td>
<td>12</td>
<td>420</td>
<td>12</td>
</tr>
<tr>
<td>$e_4$</td>
<td>12</td>
<td>192</td>
<td>0</td>
<td>240</td>
<td>192</td>
<td>192</td>
<td>168</td>
<td>168</td>
<td>12</td>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$e_5$</td>
<td>12</td>
<td>192</td>
<td>192</td>
<td>0</td>
<td>240</td>
<td>192</td>
<td>192</td>
<td>168</td>
<td>168</td>
<td>12</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>$e_6$</td>
<td>12</td>
<td>144</td>
<td>144</td>
<td>192</td>
<td>192</td>
<td>0</td>
<td>144</td>
<td>120</td>
<td>120</td>
<td>12</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>$e_7$</td>
<td>12</td>
<td>144</td>
<td>144</td>
<td>192</td>
<td>192</td>
<td>144</td>
<td>0</td>
<td>120</td>
<td>120</td>
<td>12</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>$e_8$</td>
<td>12</td>
<td>120</td>
<td>120</td>
<td>168</td>
<td>168</td>
<td>120</td>
<td>120</td>
<td>0</td>
<td>96</td>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$e_9$</td>
<td>12</td>
<td>120</td>
<td>120</td>
<td>168</td>
<td>168</td>
<td>120</td>
<td>120</td>
<td>0</td>
<td>96</td>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$e_{10}$</td>
<td>420</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$e_{35}$</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>0</td>
</tr>
</tbody>
</table>

An undirected graph is constructed by the matrix of the conflicts. In this graph, the vertices are the events, and the edges are the conflicts between them. In fact, the matrix of the conflicts is another way for a graph to be represented by an adjacency matrix. In this matrix, each element is assigned a value which corresponds to the weight of the conflict (Figure 3).
After using the heuristic algorithm (described above), each of the vertices of the graph is assigned a particular color (Figure 4).

![Figure 4](image1)

**Figure 4** An undirected colored graph with 35 vertices

The vertices colored with the same color, form a chromatic class. These vertices in separate columns can be sorted, i.e., when drawing the graph every chromatic class in a separate column is displayed (Figure 5).

![Figure 5](image2)

**Figure 5** The vertices arranged in chromatic classes

Since at the execution of the method based on local search the ordinance of the events is of importance [21], it is necessary that the chromatic classes be arranged in an appropriate manner (such as one providing the best possible result), i.e. the chromatic classes must be sorted following some criteria. The pseudo-code of the method is presented in Figure 6.

```plaintext
Initialize the weight matrix
// beginning of the graph coloring method
sort the graph vertices
// according to their degree
set color to first color
repeat
for each nonadjacent vertices do
  set vertex color to color
for all colored vertices do
  remove vertex from the graph
sort the remaining vertices
set color to next color
until all vertices are colored
// a chromatic classes sorting method
compute the chromatic classes total weight
sort the chromatic classes
// beginning of the local search method
for each event do
  if current event is fixed then next event
  for each timeslot do
    set current event to current timeslot
    if it is not possible (due to conflict)
      then go to the next timeslot
    compute solution cost and store it
    go to the next timeslot
  end // for each timeslot
  store current event in current timeslot, wherein the cost of the solution was the best and go to the next event
end // for each event
```

**Figure 6** The pseudo-code of the algorithm

In the current study, three different variants of an arrangement of the chromatic classes will be tested:
1) Depending on the color indexes (set after the performance of the heuristic algorithm for coloring the vertices of the graph).
2) Depending on the total weight of the events in a chromatic class.
3) Depending on the total count of events in a chromatic class.

The results obtained after the execution of the local search algorithm based on chromatic classes (LSCC) (for the graph presented in Figure 3) are as follows: by color index: 25.22; by total count: 14.55 and by total weight: 13.82. The best result is obtained when the chromatic classes were sorted by total weight of events in them.
4. Experimental results

The aim of the experiments is to determine the best criterion to sort chromatic classes. The experimental conditions are the following: PC with windows server 2012 R2 standard edition, 64-bit operating system with processor Intel (R) Xeon (R) E5645 at 2.40 GHz 2.39 GHz; RAM memory: 4.88 GB.

In Table 2, the results of the algorithm execution on twelve input data sets (IDS1, ..., IDS12) are shown. The chromatic classes are sorted by color index (corresponds to the direct execution of the local search method), total weight and total count. The input data sets and the results obtained by the integrated information system for university course timetabling are presented in [27, 28].

<table>
<thead>
<tr>
<th>IDS</th>
<th>E</th>
<th>S</th>
<th>L</th>
<th>A</th>
<th>Conflicts</th>
<th>Local search</th>
<th>Total count</th>
<th>Total weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>142</td>
<td>23</td>
<td>43</td>
<td>25</td>
<td>1470</td>
<td>132.05</td>
<td>110.64</td>
<td>102.83</td>
</tr>
<tr>
<td>2</td>
<td>189</td>
<td>35</td>
<td>46</td>
<td>25</td>
<td>2071</td>
<td>120.53</td>
<td>102.65</td>
<td>101.18</td>
</tr>
<tr>
<td>3</td>
<td>95</td>
<td>23</td>
<td>24</td>
<td>16</td>
<td>866</td>
<td>28.28</td>
<td>27.40</td>
<td>27.97</td>
</tr>
<tr>
<td>4</td>
<td>87</td>
<td>20</td>
<td>25</td>
<td>13</td>
<td>775</td>
<td>48.55</td>
<td>47.49</td>
<td>43.04</td>
</tr>
<tr>
<td>5</td>
<td>153</td>
<td>28</td>
<td>37</td>
<td>17</td>
<td>1904</td>
<td>89.39</td>
<td>90.27</td>
<td>80.53</td>
</tr>
<tr>
<td>6</td>
<td>121</td>
<td>28</td>
<td>39</td>
<td>16</td>
<td>1140</td>
<td>83.06</td>
<td>76.65</td>
<td>74.40</td>
</tr>
<tr>
<td>7</td>
<td>131</td>
<td>30</td>
<td>32</td>
<td>17</td>
<td>1436</td>
<td>61.17</td>
<td>59.85</td>
<td>56.98</td>
</tr>
<tr>
<td>8</td>
<td>125</td>
<td>31</td>
<td>33</td>
<td>16</td>
<td>1284</td>
<td>87.84</td>
<td>84.04</td>
<td>76.82</td>
</tr>
<tr>
<td>9</td>
<td>584</td>
<td>108</td>
<td>108</td>
<td>59</td>
<td>7754</td>
<td>299.42</td>
<td>241.14</td>
<td>228.28</td>
</tr>
</tbody>
</table>

The abbreviations used in Table 2 are as follows: E – events; S – students; L – lecturers; A – auditoriums;

The results obtained show that the LSCC algorithm generated better solutions when the chromatic classes were ranked by total weight criterion (in 9 of 12 input data sets). Only in three of the cases, the LSCC algorithm found better solutions when the sorting criterion was the total count of events in a chromatic class (Figure 7). In these three cases (IDS3, IDS10, and IDS12), the results obtained are commensurate with those achieved when sorting the chromatic classes by total weight (the differences between values are slight).

In the other nine cases, the differences between the values of the solutions found are significant. The conclusion that can be drawn is that the results obtained after the execution of the LSCC algorithm will be better if the chromatic classes are sorted according to the total weight criterion.

5. Conclusion and future work

This paper has presented a local search algorithm based on chromatic classes for the university course timetabling problem. The main idea of the algorithm was through a heuristic algorithm to specify the chromatic classes of a graph in which the events of the timetable correspond to the graph vertices and the set of the edges represents the possible conflicts between events. After then the chromatic classes of
the graph were sorted according to specific sort criteria (a total weight or a total count of events in each class) and then the local search algorithm was started. The aim of the experiments was to determine the best criterion for sorting of the chromatic classes. The obtained results showed that the algorithm generates better solutions when the chromatic classes are sorted by a total weight criterion.

The current study may be extended in the following areas:

1) Optimization of the algorithm in terms of quality, complexity, and execution time.
2) Conducting additional experiments with more input data sets for the purpose of verifying the results presented in this article.
3) Research of other criteria to sort the chromatic classes, such as the total duration of the events in each class.
4) Comparative analysis of the results obtained from the LSCC algorithm with the results obtained by other algorithms, such as GA, MA, and EGB (in terms of quality and execution time).

Acknowledgment
None.

Conflicts of interest
The authors have no conflicts of interest to declare.

References
[18] Lewis RM, Paechter B. New crossover operators for timetabling with evolutionary algorithms. International conference on recent advances in soft computing 2004 (pp. 189-95). RASC.

Velin Kralev was born in Dupnica, Bulgaria, on May 22, 1976. He received the Ph.D. degree in Informatics from South-West University, Blagoevgrad, Bulgaria in 2010. Since 2004 he has been a chief assistant professor at the South-West University, Blagoevgrad, Bulgaria. His research interests include modeling, synthesis and analysis of various algorithms, development of client-server and multi-tier computing information systems, through the use of component-oriented software technology, database design, and object-oriented programming.
Email: velin_kralev@swu.bg

Radoslava Kraleva defended her Ph.D. thesis “Acoustic-Phonetic Modeling for Children’s Speech Recognition in Bulgarian” in 2014. Most of her publications are focused into the study of children and their interaction with computer technologies, including recognition of children's speech in Bulgarian. Currently, she is working on her post-doctoral research related to the design and development of interactive mobile applications for children and the study of human-computer interface for young children. Now she is a chief assistant professor at the Department of Informatics, South-West University “Neofit Rilski”, Bulgaria. She has developed several academic courses and curricula.