Research Article

International Journal of Advanced Computer Research, Vol 7(28)
ISSN (Print): 2249-7277 ISSN (Online): 2277-7970
http://dx.doi.org/10.19101/IJACR.2016.627011

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

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. Then 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*).

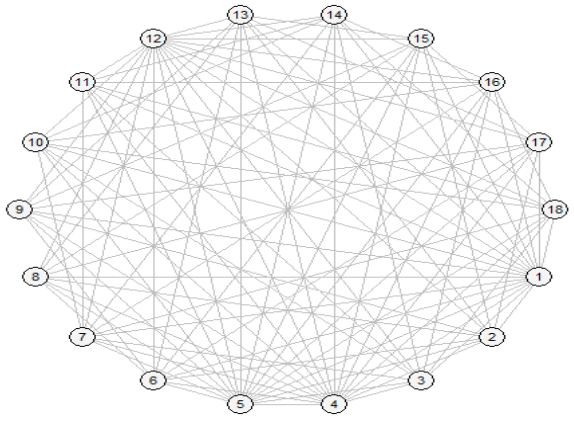

Figure 1 Undirected graph G = (V, E)

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).

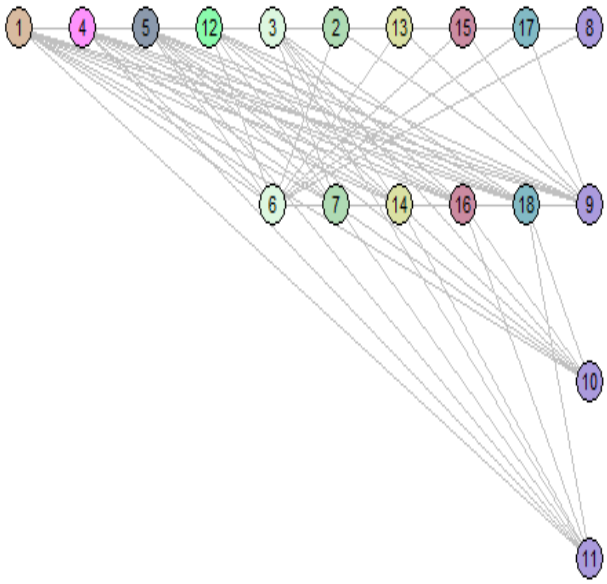

Figure 2 Colored graph G = (V, E)

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 x 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].

Table 1 A matrix of the conflicts among events

$E \backslash E$	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}	...	E_{35}
e_1	0	12	12	12	12	12	12	12	12	420	...	12
e_2	12	0	144	192	192	144	144	120	120	12	...	0
e_3	12	144	0	192	192	144	144	120	120	12	...	0
e_4	12	192	192	0	240	192	192	168	168	12	...	0
e_5	12	192	192	240	0	192	192	168	168	12	...	0
e_6	12	144	144	192	192	0	144	120	120	12	...	0
e_7	12	144	144	192	192	144	0	120	120	12	...	0
e_8	12	120	120	168	168	120	120	0	96	12	...	0
e_9	12	120	120	168	168	120	120	96	0	12	...	12
e_{10}	420	12	12	12	12	12	12	12	12	0	...	12
...	...	...	...	...	...	...	...	...	...	...	...	...
e_{35}	12	0	0	0	0	0	0	0	12	12	...	0

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*).

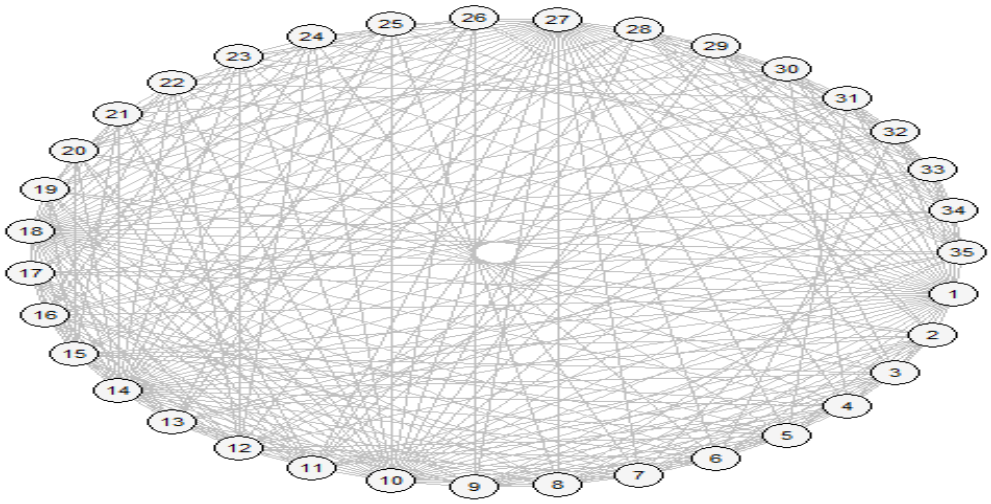

Figure 3 An undirected graph with 35 vertices

After using the heuristic algorithm (described above), each of the vertices of the graph is assigned a particular color (*Figure 4*).

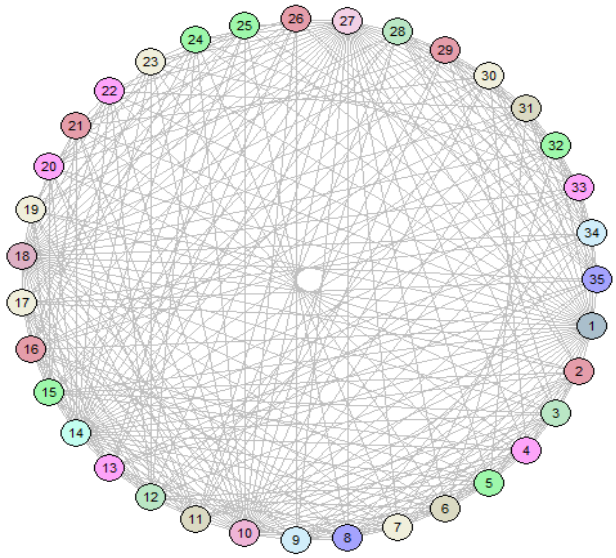

Figure 4 An undirected colored graph with 35 vertices

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*).

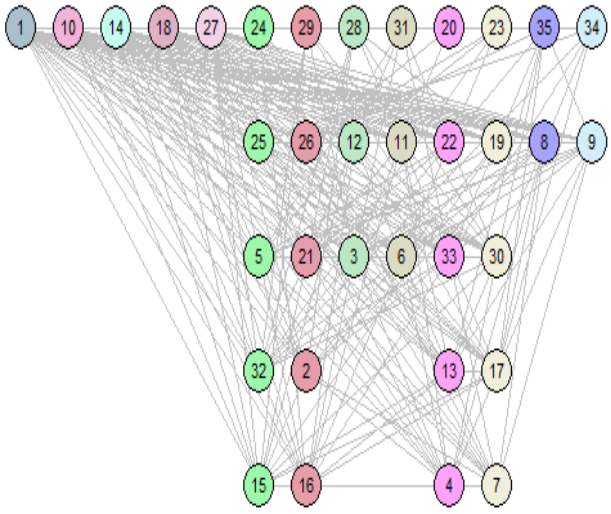

Figure 5 The vertices arranged in chromatic classes

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*.

```
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 2 Results for all input data sets

IDS	E	S	L	A	Con-flicts	Local search	Total count	Total weight
1	142	23	43	25	1470	132.05	110.64	102.83
2	189	35	46	25	2071	120.53	102.65	101.18
3	95	23	24	16	866	28.28	27.40	27.97
4	87	20	25	13	775	48.55	47.49	43.04
5	153	28	37	17	1904	89.39	90.27	80.53
6	121	28	39	16	1140	83.06	76.65	74.40
7	131	30	32	17	1436	61.17	59.85	56.98
8	125	31	33	16	1284	87.84	84.04	76.82
9	584	108	108	59	7754	299.42	241.14	228.28
10	396	89	78	42	4775	178.87	156.95	157.63
11	623	116	129	62	8550	362.35	323.37	309.41
12	496	109	110	52	5812	228.66	210.10	210.83

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.

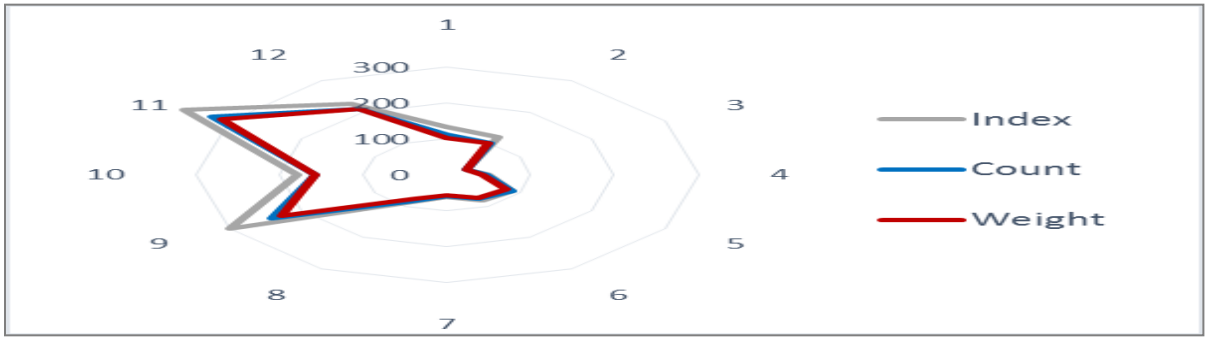

Figure 7 Results according to the sort criteria

Interpretation of the results can be made on the basis that initially the algorithm with local search finds the time slots for the events of the chromatic class with the greatest total weight, thus the time slots that have the most significant impact on the final score of the timetable are occupied by events with larger weights. Thereby, most of the soft constraints of the resources (students, lecturers, and auditoriums) are satisfied, which in turn leads to a better evaluation of the timetable.

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

[1] Gotlieb CC. The construction of class-teacher timetables. In proceedings IFIP congress 1963 (pp 73-7).
[2] Even S, Itai A, Shamir A. On the complexity of time table and multi-commodity flow problems. In annual symposium on foundations of computer science 1975 (pp. 184-93). IEEE.
[3] De Werra D. An introduction to timetabling. European Journal of Operational Research. 1985;19(2):151-62.
[4] Komijan AR, Koupaei MN. A mathematical model for university course scheduling: a case study. International Journal of Technical Research and Applications. 2015;3(19):20-5.
[5] Chen RM, Shih HF. Solving university course timetabling problems using constriction particle swarm optimization with local search. Algorithms. 2013;6(2):227-44.
[6] Kohshori MS, Abadeh MS. Hybrid genetic algorithms for university course timetabling. International Journal of Computer Science Issues. 2012; 9(2):446-55.
[7] Burke EK, Mareček J, Parkes AJ, Rudová H. Decomposition, reformulation, and diving in university course timetabling. Computers & Operations Research. 2010;37(3):582-97.
[8] Kralev V. Study and design of automated systems for allocation of resources. PhD Thesis. South-West University Press; 2010.
[9] Kang L, White GM. A logic approach to the resolution of constraints in timetabling. European Journal of Operational Research.1992;61(3):306-17.
[10] Partovi FY, Arinze B. A knowledge based approach to the faculty-course assignment problem. Socio-Economic Planning Sciences. 1995;29(3):245-56.
[11] Petrovic S, Burke E. University timetabling handbook of scheduling: algorithms, models and performance analysis. CRC Press; 2004.
[12] Burke EK, Petrovic S. Recent research directions in automated timetabling. European Journal of Operational Research. 2002;140(2):266-80.
[13] Patrick K, Godswill Z. Greedy ants colony optimization strategy for solving the curriculum based university course timetabling problem. British Journal of Mathematics & Computer Science. 2016;14(2):1-10.
[14] Elmohamed MS, Coddington P, Fox G. A comparison of annealing techniques for academic course scheduling. In international conference on the practice and theory of automated timetabling 1997 (pp. 92-112). Springer Berlin Heidelberg.
[15] Costa D. A tabu search algorithm for computing an operational timetable. European Journal of Operational Research. 1994;76(1):98-110.
[16] Abdullah S, Burke EK, Mccollum B. An investigation of variable neighbourhood search for university course timetabling. In the multidisciplinary international conference on scheduling: theory and applications (MISTA) 2005 (pp. 413-27).
[17] Soliman SE, Keshk AE. Memetic algorithm for solving university course timetabling problem. International Journal of Mechanical Engineering and Information Technology. 2015;3(8):1476-86.
[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.
[19] De Werra D. Extensions of coloring models for scheduling purposes. European Journal of Operational Research. 1996; 92(3):474-92.
[20] Kralev V. A model for the university course timetabling problem. Information Technologies & Knowledge. 2009; 3(3): 276-89.
[21] Kralev V. A genetic and memetic algorithm for solving the University course timetable problem. Information Theories & Applications. 2009; 16(3):291-9.
[22] Kralev V, Kraleva R. Variable neighborhood search based algorithm for University course timetabling problem. In proceedings of the fifth international scientific conference 2013(pp. 202-14).
[23] Kralev V, Kraleva R, Yurukov B. An event grouping based algorithm for University course timetabling problem. International Journal of Computer Science and Information Security. 2016;14(6):222-9.

[24] Burke EK, Elliman DG, Weare R. A university timetabling system based on graph colouring and constraint manipulation. Journal of Research on Computing in Education. 1994;27(1):1-18.
[25] Halldórsson MM. A still better performance guarantee for approximate graph coloring. Information Processing Letters. 1993;45(1):19-23.
[26] Welsh DJ, Powell MB. An upper bound for the chromatic number of a graph and its application to timetabling problems. The Computer Journal. 1967; 10(1):85-6.
[27] Kralev V, Kraleva R, Siniagina N. An integrated system for university course timetabling. Mathematics and Natural Science. 2009:98-104.
[28] Kralev V, Kraleva R. Web service based system for generating input data sets. Mathematics and Natural Science. 2011:49-56.

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 Departement of Informatics, South-West University "Neofit Rilski", Bulgaria. She has developed several academic courses and curricula.