Iterated Heuristic Algorithms for the Classroom Assignment Problem

Ademir Aparecido Constantino[1*], Walter Marcondes Filho[1], Dario Landa-Silva[2]

[1]*Department of Computer Science*
State University of Maringá, Brazil
[2]*ASAP Research Group*
School of Computer Science
University of Nottingham, UK

Abstract

We tackle the classroom assignment problem in a large University with the objective of minimising the total distance between all classrooms assigned to teaching activities in the same course. Additional requirements that should be satisfied include: making an efficient utilisation of the space, satisfying room preferences and complying with other administrative requirements. We present two iterated heuristic approaches, each one consisting of an iterative resolution of an assignment problem (the classical assignment problem in the first approach and the bottleneck assignment problem in the second approach) and a third algorithm based on the Variable Neighbourhood Search (VNS) meta-heuristic. We also present and discuss experimental results using real-world data from three consecutive academic sessions.

Key words: classroom assignment problem, iterated heuristic algorithm, variable neighbourhood search.

*Corresponding author: ademir@din.uem.br

1 Introduction

The classroom assignment problem in an academic institution refers to assigning classes, that meet at different timeslots, to rooms while respecting a series of operational restrictions and preferences (Carter and Covey 1992). This paper deals with a real-world classroom assignment problem from a large university involving many courses and classrooms. In our case, like in many other scenarios, the underlying course timetabling problem is solved in two phases (Carter and Laporte 1998). In the first phase, timetables are constructed for each department and each course. Since different courses can share some rooms, the availability of rooms is usually not considered in this first phase (although some courses might have priority for using certain rooms). In the second phase, rooms are assigned centrally to all courses based on the timetables produced in the first phase.

Although the classroom assignment problem is usually part of the well-known university course timetabling problem, it is also a very difficult problem and it has not been investigated on its own so extensively in the timetabling literature. Abdennadher et al. (2000) tackled this problem independently from the associated course timetabling problem and using constraint logic programming. Martinez-Alfaro et al. (1996) employed simulated annealing to assign classrooms to a large number of classes in a University in Mexico. Many times, the classroom assignment problem is tackled as part of the University course timetabling problem or the School timetabling problem (see Adriaen et al. 2006, Dammak et al 2006 and Schaefer 1999).

Carter and Tovey (1992) studied the classroom assignment problem and discussed its computational complexity. They suggested two versions of the problem, *interval* problem and *nointerval* problem, depending on how the concept of class is defined. In the *interval* problem, classes meet only once a week while in the *noninterval* (also called *multiday*) problem classes can meet more than once a week. Furthermore, when a class meets more than once a week, every meeting should occur in the same room. Following this classification, our work deals with a *noninterval classroom assignment problem*. Carter and Tovey (1992) showed

that this problem is NP-complete even for the *satisfice* case in which the problem is to find a feasible solution.

This paper is organized in 5 sections. Section 2 describes the particular classroom assignment problem tackled in this paper. Section 3 introduces some definitions and the proposed algorithms. Section 4 gives an overview of the implementation and the results. Finally, we conclude the paper in Section 5.

2 Problem Description

This work is based on the timetabling problem faced by a public higher education institution which is divided in several administrative centres and each containing related departments. Departments are responsible for offering and coordinating the various courses within their competence. Specifically, the institution is divided in 7 administrative centres and 34 departments. A total of 49 courses are on offer with approximately 4,000 subjects/sections offered to serve approximately 16,500 enrolled undergraduate students. There are 200 available classrooms for lectures plus special rooms or laboratories for practical classes. These practical classes have the special rooms assigned locally by their own departments and hence are not considered as part of the classroom assignment problem tackled here.

Despite this administrative division, the assignment of classrooms is responsibility of the institution's central administration. Students' transfers and adjustments may occur some days before the classes start. This situation makes the problem more difficult because prior assignments might need to be modified and this provokes operational administrative problems.

When assigning classrooms to classes, there are a number of restrictions and special needs for resources which hinder the classroom distribution. Several requirements must be taken into consideration:

1. Except for lectures resulting from the union of groups with practical lessons, only one lesson can be assigned in the same classroom at the same time. The classroom must be accessible to groups in which there are students with special needs.

2. Except for some subjects determined by the course, the number of students in a classroom must not be superior to its capacity.
3. Each course must have a defined geographic area for their academic activities and this serves as reference for the classroom assignment. The goal is to concentrate all classes in the same course within a geographic area of the campus.
4. Classes must be assigned to classrooms numbered according to the class year, i.e. freshman, sophomore, junior or senior year.
5. All the weekly meetings of a class should be preferably assigned to the same classroom. This facture increase the difficulty to solve the problem (*noninterval* case) as it was discussed by Carter and Tovey (1992).

The goal is to assign all the groups of all the subjects and courses to classrooms, maximizing the concentration of students of the same course within a geographical area, thus, minimizing the movement of students inside the campus while also obeying the abovementioned restrictions. Notice that requirements 1, 3 and 4 are considered preferences. In addition, some courses have preference for certain classroorns, these preferences are incorporated into the cost function (see Section 3). According to Carter and Tovey (1992) these preferences are non-monotonic (arbitrary) and increase the complexity of the assignment problem. The present work proposes the use of heuristic algorithms to solve this problem.

2.1 Definitions

There are 6 timeslots every weekday for a total of 34 timeslots per week, as shown in Table 1. Note that these 34 slots are available in each week of the entire academic year and since the allocations are the same for every week, then the solution for one week is all that is needed.

Table 1. Definition of timeslots during a week.

Period	Hour	Week					
		Mon	Tur	Wed	Thu	Fri	Sat
Morning	07:45 - 09:15	1	7	13	19	25	31
	09:30 - 11:45	2	8	14	20	26	32
Afernoon	13:30 - 15:10	3	9	15	21	27	33
	15:30 - 17:10	4	10	16	22	28	34
Night	19:30 - 21:10	5	11	17	23	29	-
	21:30 - 23:00	6	12	18	24	30	-

Consider the following notations for the indices:

m =1...M for the timeslots with M = 34,

k = 1...K for the courses,

t = 1...T_m for the groups (classes) with their timetable in timeslot m,

s = 1...S_k for the years of a course k,

l = 1...L for the classrooms.

A *classroom area* comprises of a building or an agglomerate of classrooms. Normally, the administrative centres have some preferred classroom areas for assigning classes in their courses. For each classroom area a Cartesian coordinate is given (area's central position) which is called the *area's point*.

It is desirable to assign all weekly lessons of a given group to the same classroom and also to have all the classrooms used by the same course and year within a geographic delimitation. In order to achieve this, we defined a *gravitational point* as a point in Cartesian coordinates or a scalar. The gravitational point serves as reference for the arranging of groups, years and courses within a geographic space. Three kinds of gravitational points are considered regarding the course, year and group and identified as: PGC_k, PGS_s and PGT_t, respectively. Each PGC_k corresponds to the Cartesian coordinates extracted from an image of the campus layout. The gravitational points PGS_s and PGT_t correspond to the classroom number. These values are used when attempting to arrange the years and groups following the order of the classroom numbers, i.e., a group in the initial year is assigned to the classrooms with numbers smaller than the other groups of posterior years. The gravitational points are empirically initialised. However, they are self-adjusted while the algorithms are executed.

3 Proposed Algorithms

In order to tackle the above classroom assignment problem (CAP), this section describes two iterated heuristic algorithms. The first one (CAP-A) is based on the successive resolution of the linear assignment problem whereas the second one (CAP-BA) is based on the successive resolution of the bottleneck assignment problem. A third proposed algorithm (CAP-VNS) is based on the variable

neighbourhood search (VNS) meta-heuristic and uses an initial solution obtained from the first phase of the CAP-A algorithm.

3.1. Algorithm CAP-A

This algorithm is based on the successive resolution of the linear assignment problem. The linear assignment problem is a classic linear programming problem equivalent to the minimum-cost perfect matching in a bipartite graph. For each timeslot *m* an instance of the assignment problem is created. The formulation of the assignment problem may be described as:

$$\text{Min } z_m = \sum_{t=1}^{T_m} \sum_{l=1}^{L} c_{tl}^m . x_{tl}^m$$

$$\text{s.t. } \sum_{t=1}^{T_m} x_{tl}^m = 1, \quad l = 1,..., L$$

$$\sum_{l=1}^{L} x_{tl}^m = 1, \quad t = 1,..., T_m$$

$$x_{tl}^m \in \{0,1\}, \quad t = 1,..., T_m, \quad l = 1,..., \text{L}$$

where c_{tl}^m is the cost of assigning group *t* to room *l* within timeslot *m*, and $x_{tl}^m = 1$ if group *t* is assigned to room *l* and 0 otherwise.

3.1.1 Phase 1

This phase consists of solving *M* assignment problems, one for each timeslot. Each assignment problem is defined by the square matrix $\left[c_{tl}^m\right]$; however, the number of groups may be smaller than the number of available classrooms. Thus, $T_m = T_m^{\text{Real}} + T_m^{Ficticious} = L$ will be considered, where T_m^{Real} is the actual number of existing groups and $T_m^{Ficticious}$ is the number of fictitious groups created to make the square matrix. Therefore, the cost matrix can be split in two parts, as shown in Fig 1, having their elements defined as:

> *Part I*: For $t = 1,2,\ldots, T_m^{\text{Real}}$ and $l = 1,2,\ldots,L$, we have $c_{tl}^m = f(t,l)$ where the function *f* (presented in the sequence) defines the cost of each assignment.

Part II: For t= $T_m^{Real}+1,\ldots,L$ (representing fictitious groups) and $l = 1,2,\ldots, L$, c_{tl}^m is the cost of assign a fictitious group in a classroom, in this case a large cost $c_{tl}^m = \infty$. As already mentioned above, the fictitious group is created to complement the matrix and make it square.

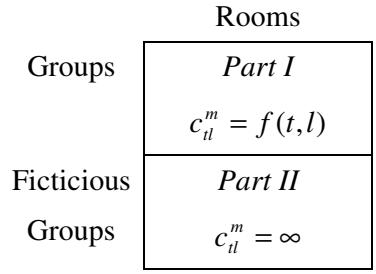

Fig. 1 Matrix basic structure

In *Part I* the function *f(t,l),* is defined as:

$$f(t,l) = \begin{cases} d_1(t,l) \text{ if } l \in SC(t) \text{ and } Size(t) \le Cap(l) \\ d_1(t,l) + p_1 \text{ if } l \in SC(t) \text{ and } Size(t) > Cap(l) \\ d_1(t,l) + p_2 \text{ if } l \notin SC(t) \text{ and } Size(t) \le Cap(l) \\ d_1(t,l) + p_1 + p_2 \text{ if } l \notin SC(t) \text{ and } Size(t) > Cap(l) \end{cases} \quad (1)$$

where:

- $d_1(t,l)$ = Euclidian distance from the PGC_k associated to group *t*, to the area's point related to room *l.*
- *SC*(*t*) is the group of classrooms with accessibility for the group *t* and their priority use is assigned to the courses from the administrative centre to which the group *t* is linked.
- *Size*(*t*) is the number of students in group *t*.
- *Cap*(*l*) is the number of students that the classroom *j* can seat.
- p_1 is the penalty applied when the classroom size does not serve the group's need. This penalty has been empirically defined as $2x10^3$.
- p_2 is the penalty defined as the biggest distance between classroom areas which belong to the same administrative centre to which the group *t* belongs. The penalty serves the purpose of forcing group *t* to be assigned to a room *l* belonging to *SC*(t).

An iteration of this phase involves the resolution of *M* assignment problems, one for each timeslot. In the first iteration, the PGC_k is empirically defined, normally a

point next to the classroom area desired for the course. For the following iterations, PGC_k is the average point estimated from the coordinates among all the classroom areas used for course k in the previous iteration. This procedure is repeated until the PGC_k of all the courses are not modified.

3.1.2 Phase 2

The purpose of this phase is to gather the groups of the same academic year in a course following the order by which the rooms are numbered, e.g. groups in the first year are in rooms with numbers smaller than the groups of the next academic year.

The structure of the cost matrix used in this phase is the same as in the previous phase, although the cost formation is slightly different, as follows:

$$f(t,l) = \begin{cases} d_1(t,l) + d_2(t,l) \text{ if } l \in SC(t) \text{ and } Size(t) \leq Cap(l) \\ d_1(t,l) + d_2(t,l) + p_1 \text{ if } l \in SC(t) \text{ and } Size(t) > Cap(l) \\ d_1(t,l) + d_2(t,l) + p_2 \text{ if } l \notin SC(t) \text{ and } Size(t) \leq Cap(l) \\ d_1(t,l) + d_2(t,l) + p_1 + p_2 \text{ if } l \notin SC(t) \text{ and } Size(t) > Cap(l) \end{cases} \quad (2)$$

where:

- $d_2(t,l) = | PGS_s - Num(l)|$, considering PGS_s the gravitational point of the year s to which the group t is related and $Num(l)$ is room number l.

An iteration of this phase also solves M assignment problems. In the first iteration $PGS_s = s$, $s = 1 \ldots S_k$, for the course k related to the group t. In the following iterations, PGS_s will be the average value of all classroom numbers allocated to the year s. This procedure is repeated until the PGS_s of all the years of every course are not modified.

3.1.3 Phase 3

The goal of this phase is to rearrange the groups gathered in phase 2 following a correspondence order for the group regarding the room numbering, e.g., if the group number 1 has been allocated to room 101, it is desirable that the group number 2 is allocated to room 102.

As in phase 2, the cost matrix structure used is the same as in phase 1, with the cost defined as follows:

$$f(t,l)=\begin{cases} d_1(t,l)+d_2(t,l)+d_3(t,l) \text{ if } \ l\in SC(t) \text{ and } Size(t)\le Cap(l) \\ d_1(t,l)+d_2(t,l)+d_3(t,l)+p_1 \text{ if } l\in SC(t) \text{ and } Size(t)> Cap(l) \\ d_1(t,l)+d_2(t,l)+d_3(t,l)+p_2 \text{ if } \ l\notin SC(t) \text{ and } Size(t)\le Cap(l) \\ d_1(t,l)+d_2(t,l)+d_3(t,l)+p_1+p_2 \text{ if } \ l\notin SC(t) \text{ and } Size(t)> Cap(l) \end{cases} \quad (3)$$

where:

- $d_3(t,l) = |PGT - Num(l)|$, considering PGT the gravitational point of the group t and $Num(l)$ is room number l.

An iteration of this phase also solves M assignment problems. In the first iteration PGT_t = t. In the following iterations, PGT_t will be the average value of the numbers of all the rooms allocated for the M modules in the previous iteration. This procedure is repeated until the PGT_t of all groups of every course are not modified.

3.2 Algorithm CAP-BA

This algorithm is equivalent to the algorithm CAP-A with the difference that the linear assignment model is replaced by the bottleneck assignment model. The bottleneck assignment problem is formulated as follows:

$$\text{Min } Z_m$$

$$\text{s.t. } \sum_{t=1}^{T_m} x_{tl}^m = 1, \quad l=1,...,L$$

$$\sum_{l=1}^{L} x_{tl}^m = 1, \quad t=1,...,T_m$$

$$c_{tl}^m x_{tl}^m \le Z_m, \quad t=1,...,T_m, \quad l=1,...,\mathrm{L}$$

$$x_{tl}^m \in \{0,1\}, \quad t=1,...,T_m, \quad l=1,...,\mathrm{L}$$

The cost matrix $\left[c_{tl}^m\right]$ is defined in the same way as for the previous algorithm. While the linear assignment model minimises the cost sum of all assignments, the bottleneck assignment model minimises the cost of the biggest assignment.

3.3. Algorithm CAP-VNS

This algorithm is based on the variable neighbourhood search (VNS) meta-heuristic, a local search procedure that explores the solution space by systematically changing the neighborhood structure (Hansen and Mladenovic,

2001). R neighbourhoods are defined for the problem in hand, N_1, N_2,...,N_R and if the current solution is not improved using a particular neighbourhood, the next neighbourhood is explored and so on.

Then, our CAP-VNS algorithm starts with a solution obtained in phase 1 of the algorithm CAP-A. Four neighbourhood structures N_r (r = 1, 2, 3 and 4) were defined. A neighbour $N_r(s)$ is obtained by exploring every timeslot for every weekday, and then choosing another assignment at random (see below).

The iterative improvement strategy used is the best descent, i.e., all solutions s" around s' are assessed, and the one giving the best improvement is selected. The evaluation function of a solution, to be minimised, is defined as:

$$f(s) = \sum_{m=1}^{34} \sum_{t=1}^{t_m} \sum_{l=1}^{L} c_{tl}^{m} . x_{tl}^{m} \qquad (4)$$

where c_{tl}^{m} is defined as in phase 3 for the algorithm CAP-A.

The neighbourhood structures N_r can be defined as:

1. N_1: for each timeslot m = 1,...,M and for each classroom area (building) used in the solution, randomly select a classroom in the classroom area and move the groups to an idle room in the same area, if possible.
2. N_2: for each timeslot m = 1,...,M and for each classroom area used in the solution, randomly select two classrooms in the classroom area and change the groups from one room to another, if possible.
3. N_3: for each timeslot m = 1,...,M and for each course with classes in module m, randomly select two classrooms used for the same course (regardless of the classroom area) and interchange all the groups between the two classrooms, if possible.
4. N_4: for each timeslot m = 1,...,M and for each year with classes in module m, randomly select two classrooms used by the year of the same course (regardless of the classroom area) and have the groups change from one room to the other, if possible.

In each iteration of the CAP-VNS algorithm, every neighbourhood is explored and the algorithm stops when there is no improvement within *3* iterations. We then follow the VNS scheme presented in Fig 2.

Initialisation. Select the set of neighbourhood structures N_r (r = 1, 2, 3 and 4); find an initial solution s; choose a stopping condition;

Repeat the following sequence until the stopping condition is met:

(1) Set $r \leftarrow 1$;

(2) *Repeat* the following steps until $r = R$:

(a) Generate a random solution s' from the r^{th} neighbourhood of s ($s' \in N_r(s)$);

(b) *Local search*. Find the best neighbour s'' of s' ($s'' \in N_r(s')$);

(c) *Move or not*. If $f(s'') < f(s)$ then $s \leftarrow s''$ and $r \leftarrow 1$; otherwise, $r \leftarrow r+1$;

Fig 2. Steps of the VNS

4 Results and Analysis

To solve the linear assignment problem, the algorithm proposed by Carpaneto and Toth (1987) was implemented which combines the Hungarian method and the Shortest Augmenting Path method. To solve the bottleneck assignment problem, the algorithm presented by Carraresi and Gallo (1984) was used.

All computational experiments were performed using a PC AMD Atlhon at 2.4 MHz, with 1 GB RAM running on Windows XP. The definition of Cartesian coordinates, used to calculate the distance between classroom areas, was based on a sketch of the institution's campus layout with a drawing scale of 2 cm = 1 m (1:50). The algorithms were tested with real data from three consecutive academic years. The characteristics of the test data used are summarised in Table 2.

Table 2. Characteristics of the test instances

Year	Number of courses	Number of rooms	Number of groups	Number of students
2006	47	170	3,927	15,270
2007	48	192	4,016	16,530
2008	49	192	3,978	16,320

Tables 3-5 present the results obtained by the proposed algorithms applied to the 3 test instances. Column **Total Cost** results from applying the objective function (equation 4) defined in Section 3.3. The number of allocations which satisfied the classroom capacity restriction is shown in the column *Favourable Allocations* (FA) and the number that did not satisfy that restriction other are shown in the column *Unfavourable Allocations* (UF). Column **Iterations** corresponds to the number of times that each phase was executed in order to reach an improved solution.

Table 3. Results for the test instance corresponding to year 2006

Algorithm	Phase	Time hh:mm:ss	Total cost	FA	UF	Iterations
CAP-A	1	00:21:08	2,015,496	3,595	332	8
	2	00:04:52	1,751,583	3,595	332	2
	3	00:17:27	1,598,637	3,595	332	3
CAP-BA	1	00:12:18	2,632,801	3,586	341	4
	2	00:07:15	2,549,259	3,587	340	2
	3	00:18:05	2,507,436	3,591	336	3
CAP-VNS	Initial Solution	00:21:08	2,015,496	3,595	332	-
	Local Search	00:26:02	1,731,850	3,595	332	3

Table 4. Results for the test instance corresponding to year 2007

Algorithm	Phase	Time hh:mm:ss	Total cost	FA	UF	Iterations
CAP-A	1	00:12:10	1,979,099	3,708	308	6
	2	00:08:15	1,733,254	3,708	308	3
	3	00:18:41	1,581,791	3,708	308	3
CAP-BA	1	00:06:11	2,589,698	3,705	311	2
	2	00:07:13	2,529,129	3,705	311	2
	3	00:22:05	2,479,152	3,706	310	4
CAP-VNS	Initial Solution	00:12:10	1,979,099	3,708	308	-
	Local Search	00:35:27	1,693,173	3,708	308	4

Table 5. Results for the test instance corresponding to year 2008

Algorithm	Phase	Time hh:mm:ss	Total Cost	FA	UF	Iterations
CAP-A	1	00:11:30	1,960,146	3,677	301	6
	2	00:09:17	1,697,943	3,677	301	3
	3	00:17:09	1,580,312	3,677	301	3
CAP-BA	1	00:06:02	2,589,036	3,666	312	2
	2	00:05:43	2,506,241	3,669	309	2
	3	00:18:22	2,493,253	3,671	307	3
CAP-VNS	Initial Solution	00:11:30	1,960,146	3,677	301	-
	Local Search	00:34:47	1,690,372	3,677	301	4

Tables 6-8 summarise the results achieved by the three algorithms proposed here together with the solution quality of the manually constructed assignments produced by the human planners. Besides the costs calculated using the objective function, these tables present the sums of the distances between the assigned rooms and the Gravitational Point of each course, measured in meters. Note that these distances are calculated based on a sketch of the campus layout. The minimum, maximum and average distances are also shown.

Table 6. Comparing results for the test instance corresponding to year 2006

Approach	Total cost	FA	UA	Total distance	Minimum Distance	Average distance	Maximum distance
CAP-A	1,598,637	3,595	332	432,855	0	158	1,680
CAP-BA	2,507,436	3,591	336	1,006,023	0	270	1,640
CAP-VNS	1,731,850	3,595	332	577,379	0	223	1,710
Manual	2,295,242	3,293	634	1,010,172	0	265	1,664

Table 7. Comparing results for the test instance corresponding to year 2007

Approach	Total cost	FA	UA	Total distance	Minimum Distance	Average distance	Maximum distance
CAP-A	1,581,791	3,708	308	506,620	0	172	1,762
CAP-BA	2,479,152	3,706	310	1,152,620	0	289	1,724
CAP-VNS	1,693,173	3,708	308	673,983	0	227	1,762
Manual	2,282,502	3,385	631	1,099,214	0	273	1,675

Table 8. Comparing results for the test instance corresponding to year 2008

Approach	Total cost	FA	UA	Total distance	Minimum distance	Average distance	Maximum distance
CAP-A	1,580,312	3,677	301	504,543	0	167	1,724
CAP-BA	2,493,253	3,671	307	1,149,892	0	286	1,724
CAP-VNS	1,690,372	3,677	301	673,057	0	225	1,724
Manual	2,281,593	3,348	630	1,098,053	0	271	1,675

It can be observed that the algorithms CAP-A and CAP-VNS achieved the best results overall. By comparing these results with those obtained manually by the institution, a very considerable improvement in the quality of the solutions can be observed, mainly with respect to the number of unfavourable allocations (UA).

5 Conclusions

In this paper, we tackled a real-world classroom assignment problem and proposed three algorithms: two iterated heuristic algorithms based on successive resolution of (linear and bottleneck) assignment problems and one algorithm based on VNS meta-heuristic. While the first and third algorithms try to minimise the total distance, the second one tries to minimise the maximum distance (min-max problem). Overall, the CAP-A algorithm performed better and reduced by more than 50% the total distance between classrooms of the same course and it also reduced considerably the number of unfavourable allocations when compared to previous manual solutions.

The computational performance of the proposed algorithms was very satisfactory regarding both solution quality and computational time. The computational time of approximately 30 to 40 minutes is quite acceptable since constructing a manual resolution for the problem can take days or weeks of work.

It is particularly important to note that CAP-A and CAP-BA are both deterministic algorithms, so, given a particular input, they always give the same solutions, while the CAP-VNS is a stochastic algorithm.

In particular, both algorithms CAP-A and CAP-BA are quite flexible with respect to the incorporation of new constraints. The required adaptation to accommodate new rules is only on the construction of the cost matrix for each assignment problem, but no change is required on the heuristic algorithms. A new hard constraint can be incorporated as an infinity cost in the cost matrix, whilst a soft constraint would be given a finite penalty cost.

Acknowledgements

This work was partly supported by the CNPq (Brazilian National Council for Scientific and Technological Development) and CAPES (Brazilian Ministry of Education).

References

Abdennadher SM, Saft S, Will S. (2000) Classroom Assignment Using Constraint Logic Programming. In: Proceedings of the Second International Conference and Exhibition on the Practical Application of Constraint Technologies and Logic Programming (PACLP 2000), Manchester.

Adriaen, M., De Causmaecker, P., Demeester, P., Vanden Berghe, G. (2006). Tackling the University Course Timetabling Problem with an Aggregation Approach. In: Proceedings of The 6th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2006), 330-335, Brno, Czech Republic.

Carpaneto, G.; Toth, P (1987). Primal-dual algorithms for the assignment problem. Discrete Applied Mathematics, vol. 18, 137-153, North-Holland.

Carraresi, P.; Gallo, G. (1984). A Multi-level Bottleneck Assignment Approach to the Bus Drivers' Rostering Problem. European Journal of Operations Research, vol. 16, 163-173.

Carter, M. W.; Tovey, C. A. (1992). When is the Classroom Assignment Problem Hard? Operations Research, vol. 40, 28-41.

Carter, M. W.; Laporte, G. (1998). Recent Developments in Practical Course Timetabling. In: The Practice and Theory of Automated Timetabling II: Selected Papers From the 2nd International Conference on the Practice and Theory of Automated Timetabling (PATAT 1997), LNCS 1408, Springer, 3-19.

Dammak, A.; Elloumi, A.; Kamoun, H. (2006). Classroom Assignment for Exam Timetabling. Advances in Engineering Software, vol. 37, no. 10, 659-666 .

Hansen, P.; Mladenovic, N. (2001). Variable Neighbourhood Search: Principles and Applications. European Journal of Operational Research, vol. 130, no. 3, 449-467.

Martinez-Alfaro, H.; Minero, J.; Alanis, G.E.; Leal, N.A.; Avila, I.G. (1996). Using Simulated Annealing to Solve the Classroom Assignment Problem. In: Proceedings of the 1st Joint Conference on Intelligent Systems/ISAI/IFIS, 370-377.

Schaefer, A. (1999). A Survey of Automated Timetabling. Artificial Intelligence Review, vol. 13, 87-127.