Simulated annealing for the uncapacitated exam scheduling problem

Meryem Cheraitia and Salim Haddadi*

LabSTIC,
8 Mai 1945 University,
Guelma, Algeria
Email: meryem.cheraitia@hotmail.fr
Email: salim.haddadi@yahoo.com
*Corresponding author

Abstract: The uncapacitated exam scheduling problem asks for a conflict-free assignment of a set of given exams into a limited number of time slots such that conflicting exams are spread as evenly as possible. In our proposed approach, using a simple graph colouring heuristic, we construct a feasible timetable to start with. Simulated Annealing is then used as an iterative heuristic for improving the spreading of the conflicting exams. Our results on the Toronto benchmark data set show that our approach provides good results and turns out competitive with 10 recent existing methods.

Keywords: metaheuristics; simulated annealing; uncapacitated exam scheduling.

Reference to this paper should be made as follows: Cheraitia, M. and Haddadi, S. (2016) 'Simulated annealing for the uncapacitated exam scheduling problem', *Int. J. Metaheuristics*, Vol. 5, No. 2, pp.156–170.

Biographical notes: Meryem Cheraitia is a PhD Student in Computer Science at the University of Guelma. Her research interests include issues related to image recognition, data mining, and operations research. She is author of a paper published in a top journal and several conference papers.

Salim Haddadi received PhD in Computer Science from Blaise Pascal University, Clermont-Ferrand, France. He is a Professor of Operations Research at the Computer Science Department, University of Guelma, Algeria. His areas of interest include combinatorial optimisation, computational complexity, and scheduling. He conducted several research projects and published several research papers, conference proceedings, and a book edited by Hermes/Lavoisier.

This paper is a revised and expanded version of a paper entitled 'Simulated annealing for the uncapacitated exam scheduling problem', presented at the *2016 International Conference on Decision Aid Sciences and Applications (DASA'16)*, Hammamet, Tunisia, 18–20 July, 2016.

1 Introduction

On one hand, timetabling is ubiquitous since it affects almost every sufficiently large institution such as educational establishments (Jaradat, Ayob and Ahmad, 2014), military barracks (Lee, Kim and Lee, 2009), hospitals (Haspeslagh et al., 2014), public transport

(Ceder, 2011), fire stations (Alutaibi, Alsubaie and Marti, 2015), and so on. On the other hand, timetabling is a very difficult and time-consuming task because it involves many constraints and a huge solution space which, although finite, grows exponentially with problem size. Timetabling is both an assignment problem and a scheduling problem, because it consists of assigning resources (rooms, students, nurses, policemen, firefighters ...) and scheduling them over time.

Exam scheduling is a special timetabling problem which is concerned with assigning a predefined set of exams into a fixed number of time slots (or periods) so as to satisfy a set of constraints. These latter may considerably vary from an establishment to another (see Merlot et al., 2003 and Thompson and Dowsland 1996 for the different kinds of examination timetabling and the most common forms of constraints). Some constraints are absolutely essential and should be fully satisfied. Such constraints are called hard. They relate to physical limitations of the real world. An invigilator, for instance, cannot be assigned to two distinct rooms at the same time. A solution satisfying the hard constraints is called a feasible timetable. Another types, deeply related to timetabling problems, are the soft constraints. These are only desirable and can be bypassed. Teachers' wishes are an example. Soft constraints cannot be completely satisfied in real-life problems because of their conflicting nature (Qu et al., 2009). Usually, the problem goal is to minimise their violation. The quality of a timetable is, however, determined by the extent to which it satisfies the soft constraints.

1.1 The uncapacitated exam scheduling problem

In this paper, we consider the uncapacitated exam scheduling problem (UESP). It is the simplest exam scheduling problem because it involves only two constraints, one hard and one soft.

Suppose that m students have to take n exams. Let $(c_{ij})_{n\times n}$ be the conflict matrix, where c_{ij} is the number of students taking both exams i and j. Let θ be the number of allowed time slots. We consider a constraint satisfaction problem, the UESP, where only one hard constraint and one soft constraint are taken into account. The hard constraint stipulates that exams with common students must be assigned to different time slots while the soft constraint requires spreading the exams over the given time slots, so as to reduce the number of students sitting exams in close proximity.

Let the integer $t_k \in \{1, \ldots, \theta\}$ specifies the time slot associated to exam k. Consider the graph $G = (X, E)$ where X is a set of n vertices, which correspond to the exams, and where E is a set of edges such that there is an edge connecting vertices i and j if $c_{ij} \neq 0$. A θ-colouring of the graph G consists of an assignment of θ colours to the vertices such that every vertex is coloured and any two adjacent vertices have different colours. A θ-colouring of the graph G thus yields a feasible exam timetable (by assigning every exam coloured with 'colour' $t \in \{1, \ldots, \theta\}$ to time slot t). The solution space of the UESP is the set of all θ-colourings of the graph G, which is finite although it may increase exponentially with problem size. The goal of the UESP is to select a θ-colouring which satisfies the soft constraint. One clever way for dealing was introduced by Carter, Laporte and Lee (1996) and was subsequently widely adopted: find a θ-colouring of the graph G, so as to minimise

$$\frac{1}{m}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c_{ij} \times p\left(t_i, t_j\right) \tag{1}$$

where

$$p\left(t_i, t_j\right) = \begin{cases} 2^5/2^{|t_j - t_i|} & \text{if } 1 \leq |t_j - t_i| \leq 5 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The objective function (1) represents in some sense the soft constraint violation. The proximity costs $p\left(t_i, t_j\right)$ assess the quality of the timetable by penalising the proximity of two conflicting exams. If the conflicting exams i and j are assigned to adjacent time slots t_i and t_j (such that $|t_j - t_i| = 1$), then the penalty is the higher, 16. If there is one (resp. two, three, four) free day(s) between the two time slots t_i and t_j ($|t_j - t_i| = 2$, resp. $3, 4, 5$), then the penalty is 8, resp. $4, 2, 1$. In any other case, the penalty $p\left(t_i, t_j\right)$ is 0.

Stated this way, the UESP becomes clearly a combinatorial optimisation problem, because if S is the set of all θ-colourings, the UESP is of the general form: Find $s^* \in S$ such that $f\left(s^*\right) = min_{s \in S} f\left(s\right)$ where $f\left(s\right)$ is the objective function in (1). The UESP is computationally intractable, since it contains as special case the decision problem of whether there exists a feasible timetable which is the well-known NP-complete θ-colouring problem (Malaguti and Toth, 2010).

1.2 The proposed approach

Since the UESP is intractable, a practical and reasonable approach is to employ a heuristic capable of finding near optimal timetables in an acceptable computing time. Our heuristic can be seen as a two-phase method: Using a simple graph colouring heuristic, the first phase begins by constructing a feasible timetable. In the second phase, simulated annealing (SA) is used for improving the timetable quality (minimising the soft constraint violation).

SA, as an intelligent LS method, is an attractive option for tackling complex optimisation problems. Its main advantage is that it has the potential to yield optimal or very near optimal solutions, and its most serious shortcoming, however, is that it a time-consuming process.

Unfortunately, there are only a few papers, of which we are aware, using SA and dealing with examination timetabling (Burke, Newall and Petrovic, 2004; Dun, Wang and Shao, 2014; Merlot et al., 2003; Thompson and Dowsland, 1996, 1998), and only one of them considers the UESP (Merlot et al., 2003).

1.3 Our contributions

The effectiveness of SA depends heavily on a suitable choice of the parameters which control the search. The two crucial parameters are the neighbourhood structure and the cooling schedule (that is the way in which the temperature is adjusted). This is exactly where lies our contribution.

- In this paper, we propose three neighbourhood structures. The first, called Exam Shifting, considers an exam and tries to assign it to another time slot. The second, called Interchange, considers swapping two different exams. The third, Kempe Chain, tries to swap subsets of exams in two time slots. Interchange is well known and Kempe Chain is widely used (Burke, Newall and Petrovic, 2004; Gogos, Alefragis and Housos, 2012; Merlot et al., 2003; Thompson and Dowsland, 1998). It is well known that Kempe Chain is the best choice (Dowsland and Thompson, 2012; Merlot et al., 2003). However, Exam Shifting turns out very appealing. Since

Kempe Chain is time consuming, to allow a large number of iterations or moves, it is explored only 20% of the time, while the two others are considered 40% of the time each. To the best of our knowledge, this is the first time the first neighbourhood is proposed for search in the context of examination timetabling. Furthermore, the authors usually consider a single neighbourhood for search. We know of only a single paper (Aycan and Ayav, 2009) which considers simultaneously two neighbourhoods. The authors report that exploring two neighbourhoods simultaneously is better than searching them separately. In this paper, we consider three different structures simultaneously.

- Following the recommendations in (Dowsland and Thompson, 2012; Thompson and Dowsland, 1995, 1996, 1998), where several empirical studies are conducted on successful implementations of SA, we adopted a robust and very slow geometric cooling schedule, but spending less time exploring the neighbourhoods at high temperature, and allowing more time progressively.

1.4 Outline of the paper

The document is organised as follows. Section 2 quickly reviews the related literature. Basic facts about the SA algorithm are outlined in Section 3. Section 4 details the proposed method, while Section 5 is devoted to the computing experiments, where our results are reported and compared with existing methods. The last section 6 concludes the article.

2 Literature overview

The UESP is one of the most studied problems. Numerous surveys were published. Some of the most interesting are due to Carter, Laporte and Lee (1996), Petrovic and Burke (2004) and Qu et al. (2009). A wide variety of approaches, dealing with almost all of the paradigms of operations research, have been proposed. The early works focus on exact approaches. Since the computation time required for finding an exact solution increases exponentially with the instance size, little effort is put forth on them nowadays (MirHassani, 2006; Woumans et al., 2016).

At the contrary, approximate methods provide good exam timetables in a reasonable amount of time. Petrovic and Burke (2004) categorise them into seven classes, from the earliest to the latest: sequential methods, cluster methods, constraint-based methods, metaheuristics, multi-criteria methods, case-based reasoning, and hyper-heuristics. Sequential methods are those which model the problem in terms of graph colouring. These earliest heuristics are, nowadays, hybridised with other techniques (Qu and Burke, 2009). Recent research includes the use of adaptive ordering techniques combined with graph colouring heuristics, where a heuristic is used to measure the difficulty of scheduling a particular exam, and where difficult exams are scheduled first (Abdul-Rahman et al., 2014; Sabar et al., 2012).

A metaheuristic is a high-level algorithm designed to drive a simple low-level heuristic for finding good solutions to an optimisation problem, by sampling the solution set which is too large to be completely searched. A metaheuristic starts either with an initial solution (local search based) or an initial set of solutions (population based), followed by an improvement procedure. A recent overview is given in (Nesmachnow, 2014). With the birth of metaheuristics, a large step is made towards solving combinatorial optimisation problems. A large number of metaheuristic approaches have been considered. Examples of these approaches include variable neighbourhood search (Abdullah et al., 2007), local search (Caramia, Dell'Olmo and Italiano, 2001), great deluge (Abdullah, Turabieh and McCollum, 2009), particle swarm optimisation (Ahandani et al., 2012), memetic techniques (Al-Betar, Khader and Abu Doush, 2014), ant or bee colony (Alzaqebah and Abdullah, 2015; Bolaji et al., 2015), genetic algorithm (Arogundade, Akinwale and Aweda, 2010; Dun, Wang and Shao, 2014), tabu search (Pais and Amaral, 2012), GRASP (Casey and Thompson, 2003), fuzzy reasoning (Asmuni, Burke and Garibaldi, 2005), and SA (Burke, Newall and Petrovic, 2004; Dun, Wang and Shao, 2014; Gogos, Alefragis and Housos, 2012; Merlot et al., 2003; Thompson and Dowsland, 1998).

Multi-criteria methods permit more flexibility in treating the soft constraints. Examples of these approaches are proposed in Petrovic and Burke (2004). Case-based reasoning methods are discussed in Petrovic and Burke (2004). The trend is towards the hyper-heuristics which are more general and more powerful methods. A comprehensive survey is proposed in Pillay (2014). Specific hyper-heuristic approaches can be found in Burke and Newall (2004); Burke et al. (2007); Burke, Qu and Soghier (2014); Pillay and Banzhaf (2009); Sabar et al. (2012).

3 An outline of the SA algorithm

Annealing is the physical process of heating a solid until its melting point and then cooling it. Kirkpatrick, Gellat and Vecchi (1983) discovered the analogy between this process of annealing and combinatorial optimisation, by noting the correspondence between, respectively, the system states, the changes of state, the temperature and the energy of the former, the feasible solutions, the moves, the control parameter, and the objective function of the latter. They then proposed an algorithm for combinatorial optimisation, which they called *Simulated Annealing*, which emulates the annealing process. Since its introduction, the SA algorithm has had a large record of successful applications in a broad range of areas (Suman and Kumar, 2006). This section recalls some basic facts. In more detail, the reader is referred to the excellent tutorials of Dowsland and Thompson (2012) and Monticelli, Romero and Asada (2008).

Consider a combinatorial optimisation problem with solution space S and objective function $f : S \longrightarrow \mathbb{R}$. Any element $s \in S$ is a solution. The neighbourhood of a given solution s is the set $N(s)$ of all the solutions that can be obtained from s by 'small changes' (the precise definition of a neighbourhood is thus problem-specific). Given $s \in S$, any $s' \in N(s)$ is called neighbour of s, and a transition from s to s' is referred as move.

Since the SA algorithm is a smart version of the well-known local search (LS), perhaps it is better understood if we begin by outlining the latter. In LS, we begin with an initial solution $s_{initial} \in S$ which is temporarily considered as the current best solution ($s_{best} = s_{initial}$). At any iteration, the actual best solution s_{best} is examined for improvement by considering only the solutions which lie in its neighbourhood. If there exists $s \in N(s_{best})$ such that $f(s) < f(s_{best})$ then we got an improvement. In this case, the solution s becomes the best current one ($s_{best} = s$) and the method iterates. Otherwise, $f(s) \geq f(s_{best})$ for all $s \in N(s_{best})$, and LS terminates. Note that the algorithm just described is a greedy LS, and that non-greedy variants do exist. The strategy which consists of accepting only improving moves has a serious drawback: the search ends trapped in a local minimum solution (due the myopic nature of the search).

The SA algorithm, with many other metaheuristics, has the same guiding principle, except that it can escape from local minimum solutions by allowing the search to accept non-improving or uphill moves. However, the search must be done carefully in order to ensure the convergence to a global minimum. In the case of SA, the search is controlled in a manner inspired by the annealing process. The basic algorithm is presented in Algorithm 1. In Line 6, $nb\,(k)$ is the number of transitions at temperature T_k. In Line 12, α is the reduction function or rate of cooling. At any iteration, the algorithm proceeds in this way: a neighbour of the current best solution is randomly selected; if it is better, the move is accepted; otherwise (Line 10) we accept the move with probability $\exp\left(-\delta/T_k\right)$. As it can be seen, the probability of accepting an uphill move depends on two factors, the temperature T_k and the level of increase δ in the cost function f. For these reasons, at the same temperature, large increases are rejected more frequently than small ones. Furthermore, the probability of accepting uphill moves will diminish steadily.

From concepts in mechanics and Markov chains, theory establishes that the SA algorithm converges to a global minimum solution in an exponential number of iterations as long as the neighbourhood configuration satisfies the reachability property (we can reach any solution from any other by a finite sequence of moves). Empirical studies, for their part, note that the two most critical issues of any SA implementation, are the neighbourhood configuration and the cooling schedule, which is the strategy used to control the algorithm.

Algorithm 1 Basic SA algorithm in pseudocode

```
1   **Select** $s_{initial} \in S$
2   $s_{accepted} \longleftarrow s_{initial}$
3   $s_{best} \longleftarrow s_{initial}$
4   $k \longleftarrow 0$ // Counts the number of temperature changes
5   Let $T_0$ be the initial temperature
6   **While** (stopping criterion not satisfied)
7       **For** $it = 1, \cdots, nb(k)$
8           **Choose** $s \in N\left(s_{accepted}\right)$
9           $\delta \longleftarrow f\left(s\right) - f\left(s_{accepted}\right)$
10          **If (** $\delta < 0$ **)** $s_{accepted} \longleftarrow s$
11          **else if** ( random$[0,1[ < \exp\left(-\delta/T_k\right)$ ) $s_{accepted} \longleftarrow s$
12          **if** ( $f\left(s_{best}\right) > f\left(s_{accepted}\right)$ ) $s_{best} \longleftarrow s_{accepted}$
13      **End For**
14      $T_{k+1} \longleftarrow \alpha\left(T_k\right)$
15      $k \longleftarrow k+1$
16  **End While**
17  **Output** $s_{best}$
```

4 The proposed method

This section presents our proposed approach. Details are given to make the algorithm reproducible (Algorithm 2). Recall that the solution space S of the UESP is the set of all legal θ-colourings. The data structure adopted is quite simple. A solution is encoded as a θ-array $[P_1, \ldots, P_\theta]$ where entry P_i is a pointer to a linked list whose nodes represent the exams belonging to time slot t_i. The family $P_1, \ldots, P_\theta$ constitutes a partition of the set of exams, and any P_j (a stable set of the graph G) stands for the jth timeslot whose elements are the exams assigned to the jth timeslot (vertices coloured with colour j).

4.1 *Initial solution*

The initial solution (see Line 1 of the pseudocode in Algorithm 2), which may be any legal θ-colouring, is obtained by employing a well-known and widely used graph colouring heuristic called "Saturation Degree with Backtracking" to the graph referred to in the introduction (see Carter, Laporte and Lee, 1996; Ahandani et al., 2012; and Burke and Newall, 2004 for a comprehensive presentation, and Malaguti and Toth (2010) for more general graph colouring methods and applications).

Algorithm 2 Pseudocode of our method

```
1   Compute an initial feasible exam timetable $s_{initial}$
2   $s_{accepted} \longleftarrow s_{initial}$
3   $s_{best} \longleftarrow s_{initial}$
4   $k \longleftarrow 0$
5   Let $T_0$ and $T_f$ be, respectively, the initial and final temperature
6   while ( $T_k \leq T_f$ and elapsed time $< 7200$ )
7       for $it = 1, \cdots, nb(k)$
8           if ( random$[0,1[\ \leq 0.2$ ) choose $s \in N_1\,(s_{accepted})$
9           else if ( $0.2 <$ random$[0,1[\ \leq 0.6$ ) choose $s \in N_2\,(s_{accepted})$
10          else if ( $0.6 <$ random$[0,1[\ \leq 1$ ) choose $s \in N_3\,(s_{accepted})$
11          $\delta \longleftarrow f\,(s) - f\,(s_{accepted})$
12          if ( $\delta < 0$ ) $s_{accepted} \longleftarrow s$
13          else if ( random$[0,1[\ < \exp\,(-\delta/T_k)$ ) $s_{accepted} \longleftarrow s$
14          if ( $f\,(s_{best}) > f\,(s_{accepted})$) $s_{best} \longleftarrow s_{accepted}$
15      end for
16      $k \longleftarrow k + 1$
17      $T_k \longleftarrow \alpha\,(T_k)$
18  end while
19  output $s_{best}$
```

Saturation Degree with Backtracking heuristics consists of colouring first the exams with the smallest number of available legal colours (time slots). Ties are broken by 'Largest Degree First' strategy. This heuristic is dynamic in the sense where the number of legal colours available for the uncoloured vertices needs to be recalculated after every vertex colouring. If no valid time slots are available for assigning some exam, Backtracking attempts to make room for it by freeing exams already assigned. Rules should be imposed to avoid cycling. Burke and Newall (2004) point the higher effectiveness of this heuristic in producing good quality starting timetables.

4.2 *Neighbourhood configurations*

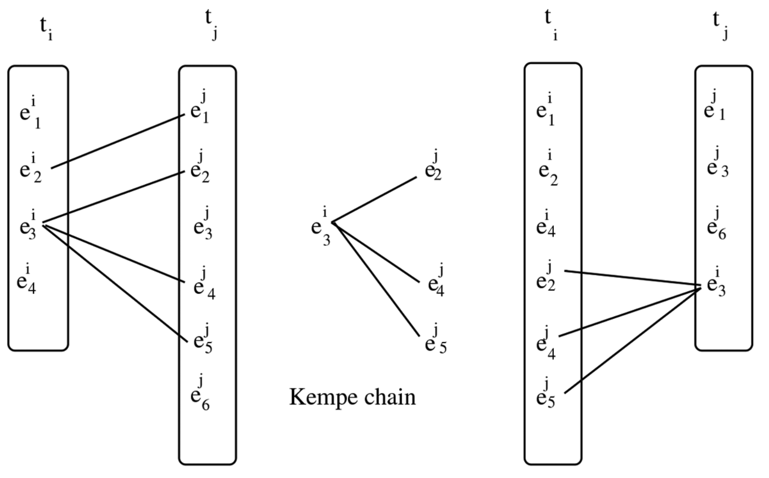
A crucial concern in the design of any local search heuristic is the manner with which the neighbourhood is defined. In this paper, we have three proposals.

1. Exam shifting: a neighbouring solution is obtained from the current by shifting a given exam. This means that the exam is ejected from the timeslot it belongs to and reassigned to another one. Unfortunately, this is not always possible. For exam k to incorporate timeslot t_j, we must ensure that $c_{ki} = 0$ for all $i \in t_j$, which means that exam k is compatible (or non-conflicting) with all the exams in timeslot j. Since there are n exams and θ time slots, clearly the size of this neighbourhood is $O(n \times \theta)$.

2. Time slot interchange: a neighbour is obtained by interchanging two time slots. Obviously, this is always possible without affecting the feasibility. The neighbourhood size is $O(\theta^2)$ since there are $\theta(\theta - 1)/2$ ways of choosing two distinct time slots.

3. Kempe chain: consider two arbitrary time slots t_i and t_j, and consider the bipartite graph $B = (t_i, t_j, E)$ where E is the set of edges connecting any two conflicting exams, one belonging to t_i and the other to t_j. Kempe chain refers to any connected component $C = C_i \cup C_j$ in the bipartite graph B where $C_i \subset t_i$ (resp. $C_j \subset t_j$). In this neighbourhood structure, a neighbour is obtained by swapping the sets C_i and C_j. An example is given in Figure 1 where $C_i = \{e_3^i\}$ and $C_j = \{e_2^j, e_4^j, e_5^j\}$. It is easy to see that this move preserves feasibility. The size of the neighbourhood is instance-dependent. There are $\theta(\theta - 1)/2$ manners of choosing two time slots, but, given the latter, the number of connected components in the underlying bipartite graph may be exponential in the cardinality of $t_i \times t_j$. So, Kempe chain defines large neighbourhoods. It is well established that the larger the neighbourhood, the better is the incumbent, but also the longer it takes to explore it. As a matter of fact, the research conducted in Thompson and Dowsland (1998) concludes that Kempe chain is the best choice (Al-Betar, Khader and Abu Doush, 2014).

Figure 1 Example of Kempe chain move



(a) Configuration before move. (b) Configuration after move.

While the two neighbourhood structures based on time slots interchange and Kempe chain satisfy the reachability property, unluckily, we do not have any formal proof of whether or not the first neighbourhood structure, exam shifting, fulfils the property. In the negative, a bias may be introduced in the search which can prevent the SA algorithm from converging to a global optimum.

Lines 8–14 of Algorithm 2 constitute an iteration or a move, either accepted or not. Line 14 updates the best timetable found so far, which will be an output at the end of the method. Due to the overhead caused by the search, storage, and update of Kempe chains, we adopted the following strategy in Lines 8–10 of the pseudocode of the SA algorithm. We flip a 'three-sided' coin, sides 2 and 3 with probability 0.4 each, and side 1 with probability 0.2. Whichever side appears (side 1 corresponds to Kempe chain neighbourhood), we choose the corresponding neighbourhood for search.

5 Experimental results

5.1 Experimental design

The proposed method is coded in C and run on an Intel Core I3 (2.4 GHz). The code is tested on the Toronto benchmark data set. The latter consists of 13 instances which are freely available (http://www.asap.cs.nott.ac.uk/resources.data.shtml). One of the instances, PUR93, is usually omitted for two reasons: it has a very large size (Müller, 2014), and very little concerning it is reported in the literature. The characteristics of the remaining twelve instances are presented in the first five columns of Table 1. The first column gives a name to the instance. The second and third columns present, respectively, the number of exams and students. The fourth provides the density (in percentage) of the conflict matrix, and the fifth the number of allowed time slots.

5.2 Parameter settings

Several runs are first performed for fine-tuning the cooling parameters. The initial temperature (Line 5 of Algorithm 2) is set to 1. At this value, about 80% in average of the moves are accepted. Then the cooling begins until it reaches a final temperature of 0.1. The reduction function, or rate of cooling, is determined by following the recommendation in Thompson and Dowsland (1995, 1996, 1998). We adopted a geometric cooling schedule by allowing $\alpha(T) = 0.99 \times T$ in Line 17 of Algorithm 2. This leads the algorithm to perform about 230 temperature changes since $0.1 \simeq 0.99^{230}$.

It is also recommended in Dowsland and Thompson (2012) to spend less time exploring the neighbourhood at high temperature where most moves are accepted. The better is to allow more time progressively. In our implementation, we adopted the following function $nb(k) = 100 \times 1.02^k$ for the number of transitions at temperature T_k (Line 7 of Algorithm 2). So, at the beginning, at temperature 1, 100 iterations are allowed, and at the final temperature, the number of iterations is about 9,500. This strategy, as confirmed by the computational experience, is rather demanding in terms of computational effort (Monticelli, Romero and Asada, 2008).

Two stopping criteria are imposed. The algorithm finishes either when the temperature attains its lowest value 0.1, or when the computing time reaches a time limit of 7,200 s.

5.3 Results

The initial solution mean values (over 10 runs) are reported in the sixth and seventh columns of Table 1, labelled ‘Cost’ and ‘Time’, which refer, respectively, to the objective function value (1) of the initial timetable obtained by using the saturation degree with backtracking method, and the average computing time. We see that the initial timetable is obtained very quickly, in less than half a second, but has a very poor quality (large cost value) because it is obtained without caring the soft constraint.

Our method is run 10 times on every instance, each time with a new initial solution. The costs of the best solutions found are recorded in the last column of Table 1. As it can be seen, there is a large deviation (of about 50%) of the best solution from the initial solution. We observe, in early iterations, at high temperature, several successive improvements in the objective function value. At the contrary, in final iterations, at low temperature, most moves are rejected, and we get sporadic improvements.

The computing times always reach the time limit of 7,200 s for every run on every instance. This is well explained. Since the cooling schedule is very slow, a huge number of iterations or moves (either accepted or not) $100 \times \sum_{k=0}^{k=230} 1.02^k$ are allowed. Without imposing a time limit, the computing times would be very large.

Table 1 Characteristics of the benchmarks, initial and final solutions

Characteristics of the benchmark instances					Initial solution		Best solution
Instance	Exams	Students	Density	Timeslots	Cost	Time	Cost
CAR91	682	16,925	0.13	35	11.01	0.68	5.26
CAR92	543	18,419	0.14	32	8.39	0.48	4.34
EAR83	189	1,108	0.27	24	57.18	0.17	34.17
HEC92	80	2,823	0.42	18	17.63	0.21	10.43
KFU93	461	5,349	0.06	20	44.88	0.28	14.36
LSE91	381	2,726	0.06	18	26.21	0.23	11.25
RYE93	486	11,483	0.07	23	30.96	0.31	9.47
STA83	138	549	0.14	13	182.49	0.23	157.13
TRE92	261	4,360	0.18	23	14.67	0.20	8.47
UTA92	638	21,329	0.12	35	6.60	0.48	3.56
UTE92	184	2,749	0.08	10	48.98	0.21	25.71
YOR83	180	919	0.29	21	52.36	0.21	36.92

5.4 Comparison with 10 existing methods

To evaluate the performance of our algorithm (called CH), we compare it with 10 recent heuristics. These methods are named after their authors:

- ABBMO (Abdul-Rahman et al., 2014) proposes a hyper-heuristic constructive approach based on adaptive decomposition and ordering of four low level graph colouring heuristics.
- AVZG (Ahandani et al., 2012) considers a hybrid particle swarm optimisation embedded into a hyper-heuristic structure, where combination of mutation, recombination operator, and graph colouring heuristics are used to update position of particles.

- BKAA (Bolaji et al., 2015) presents a hybrid artificial bee colony algorithm.
- BMMPQ (Burke et al., 2007) is a multi-stage hyper-heuristic where tabu search is used for selecting between graph heuristics.
- BQS (Burke, Qu and Soghier, 2014) is another hyper-heuristic which hybridises low-level graph colouring heuristics.
- CDI (Caramia, Dell'Olmo and Italiano, 2001) is another hybrid algorithm.
- PA (Pais and Amaral, 2012) suggests a fuzzy inference rule-based system for tuning some parameters of Tabu Search.
- PB (Pillay and Banzhaf, 2009) presents a hyper-heuristic based on hierarchical combination of heuristics.
- QB (Qu and Burke, 2009) is yet another hyper-heuristic which hybridises low-level graph colouring heuristics.
- SAQK (Sabar et al., 2012) deals with another graph colouring constructive hyper-heuristic.

The authors of all of these methods (except CDI where only five runs are allowed) pick the best solution out of 10 runs for every instance. Table 2 shows the cost of the best solution found by every method (taken from the literature). The mark – in a cell means that the corresponding value is not reported. The best value among the eleven is boldfaced. It can be seen that our method never finds the best. The two methods BKAA and CDI seem the best qualified from this side. But if we consider the last row of Table 2 which reports the average deviation from best (which is, in our opinion, a primary parameter for comparison), our method takes the third rank behind the methods BKAA and CDI. Our method is competitive with BKAA, but the two are dominated by CDI. This is due to the fact that the method CDI finds two extraordinarily good solutions for the instances EAR83 and RYE93.

Table 2 Comparison of our approach with existing methods from accuracy point of view

	ABBMO	AVZG	BKAA	BMMPQ	BQS	CDI	PA	PB	QB	SAQK	CH
CAR91	5.17	5.22	**4.22**	5.41	5.19	6.6	5.46	4.97	5.16	5.14	5.26
CAR92	4.74	4.67	5.00	4.84	4.31	6.0	4.57	4.28	**4.16**	4.70	4.34
EAR83	40.91	35.74	34.08	38.19	35.79	**29.3**	33.50	36.86	35.86	37.86	34.17
HEC92	12.26	10.74	10.32	12.72	11.19	**9.2**	10.52	11.85	11.94	11.90	10.43
KFU93	15.85	14.47	13.91	15.76	14.51	**13.8**	14.05	14.62	14.79	15.30	14.36
LSE91	12.58	10.76	11.04	13.15	10.92	**9.6**	–	11.14	11.15	12.33	11.25
RYE93	10.11	9.95	9.18	–	–	**6.8**	9.11	–	–	10.71	9.47
STA83	158.12	157.10	**157.04**	157.38	157.18	158.2	157.29	158.19	158.33	160.12	157.13
TRE92	9.30	8.47	8.38	8.85	8.49	9.4	8.71	8.48	8.60	**8.32**	8.47
UTA92	3.65	3.52	**3.40**	3.88	3.44	3.5	3.71	**3.40**	3.42	3.88	3.56
UTE92	27.71	25.86	25.80	31.65	26.70	**24.4**	25.18	28.88	28.30	32.67	25.71
YOR83	43.98	38.72	36.53	40.13	39.47	**36.2**	39.08	40.74	40.24	40.53	36.92
	3.16	1.57	1.04^2	2.94	1.60	0.55^1	1.72	2.16	2.03	3.08	1.22^3

Table 3 ranks the 11 methods from the point of view of the best solution found. The last row of the same table gives an average rank for the compared methods. Once again, we can see from this table that our method is competitive with the two methods BKAA and CDI.

The two parameters of comparison (average deviation from best and average rank) are thus consistent and provide almost the same conclusion: the three best methods are BKAA, CDI, and CH, the first two slightly dominating the last.

Table 3 Average rank among eleven heuristics

	ABBMO	AVZG	BKAA	BMMPQ	BQS	CDI	PA	PB	QB	SAQK	CH
CAR91	5	7	1	9	6	11	10	2	4	3	8
CAR92	8	6	10	9	3	11	5	2	1	7	4
EAR83	11	5	3	10	6	1	2	8	7	9	4
HEC92	10	5	2	11	6	1	4	7	9	8	3
KFU93	11	5	2	8	6	1	3	7	9	10	4
LSE91	9	2	4	10	3	1	–	5	6	8	7
RYE93	6	5	3	–	–	1	2	–	–	7	4
STA83	7	2	1	6	4	8	5	9	10	11	3
TRE92	10	3	2	9	6	11	8	5	7	1	3
UTA92	8	6	1	10	4	5	9	1	3	10	7
UTE92	7	5	4	10	6	1	2	9	8	11	3
YOR83	11	4	2	7	6	1	5	10	8	9	3
	8.58	4.58	2.92[1]	9.00	5.09	4.42[2]	5.00	5.91	6.55	7.83	4.42[2]

6 Conclusion

In this article, we deal with a computationally intractable problem, the UESP, for which a metaheuristic is designed. Begin with a first placement obtained by using a simple graph colouring heuristic, the SA algorithm is used for iteratively improving its quality (spreading the exams over the periods). Three distinct neighbourhoods are proposed concurrently for search, and a robust geometric cooling schedule is designed for controlling the temperature change. The results obtained on a set of benchmark instances and the comparison from two points of view (average deviation from best and rank) with 10 recent methods show that our method is competitive with the state-of-the-art.

The main conclusion that can be drawn from this computational study is that the SA algorithm, although time consuming, still remains a good tool, when well controlled, for tackling hard combinatorial optimisation problems.

Acknowledgement

The authors wish to thank the anonymous referees for their fruitful comments which contribute to greatly improve the clarity and readability of the manuscript.

References

Abdullah, S., Ahmadi, S., Burke, E.K. and Dror, M. (2007) 'Investigating Ahuja-Orlin's large neighbourhood search approach for examination timetabling', *OR Spectrum*, Vol. 29, pp.351–372.

Abdullah, S., Turabieh, H. and McCollum, B. (2009) 'A hybridisation of electromagnetic-like mechanism and great deluge for examination timetabling problems', *Lecture Notes in Computer Science*, Vol. 5818, pp.60–72.

Abdul-Rahman, S., Burke, E.K., Bargiela, A., McCollum, B. and Ozcan, E. (2014) 'A constructive approach to examination timetabling based on adaptive decomposition and ordering', *Annals of Operations Research*, Vol. 218, pp.3–21.

Ahandani, M.A., Vakil Baghmisheh, M.T., Zadeh, M.A.B. and Ghaemi, S. (2012) 'Hybrid particle swarm optimization transplanted into a hyper-heuristic structure for solving examination timetabling problem', *Swarm Evolutionary Computation*, Vol. 7, pp.21–34.

Al-Betar, M.A., Khader, A.T. and Abu Doush, I. (2014) 'Memetic techniques for examination timetabling', *Annals of Operations Research*, Vol. 218, pp.23–50.

Alutaibi, K., Alsubaie, A. and Marti, J. (2015) 'Allocation and scheduling of firefighting units in large petrochemical complexes', in *International Conference on Critical Infrastructure Protection*, Springer International Publishing, Cham, pp.263–279.

Alzaqebah, M. and Abdullah, S. (2015) 'Hybrid bee colony optimization for examination timetabling problems', *Computers and Operations Research*, Vol. 54, pp.142–154.

Arogundade, O.T., Akinwale, A.T. and Aweda, O.M. (2010) 'A genetic algorithm approach for a real-world university examination timetabling problem', *International Journal of Computer Applications*, Vol. 12, pp.1–4.

Asmuni, H., Burke, E.K. and Garibaldi, J.M. (2005) 'Fuzzy multiple ordering criteria for examination timetabling', *Lecture Notes in Computer Science*, Vol. 3616, pp.334–353.

Aycan, E. and Ayav, T. (2009) 'Solving the course scheduling problem using simulated annealing', *Proceedings of the IEEE International Advance Computing Conference*, Patiala, India, pp.462–466.

Bolaji, A.L., Khader, A.T., Al-Betar, M.A. and Awadallah, M.A. (2015) 'A hybrid nature-inspired artificial bee colony algorithm for uncapacitated examination timetabling problems', *Journal of Intelligent Systems*, Vol. 24, pp.37–54.

Burke, E.K., Bykov, Y., Newall, J.P. and Petrovic, S. (2004) 'A time-predefined local search approach to exam timetabling problems', *IIE Transactions*, Vol. 36, pp.509–528.

Burke, E.K., McCollum, B., Meisels, A., Petrovic, S. and Qu, R. (2007) 'A graph-based hyper-heuristic for educational timetabling problems', *European Journal of Operational Research*, Vol. 176, pp.177–192.

Burke, E.K. and Newall, J.P. (2004) 'Solving examination timetabling problems through adaption of heuristic orderings', *Annals of Operations Research*, Vol. 129, pp.107–134.

Burke, E.K., Qu, R. and Soghier, A. (2014) 'Adaptive selection of heuristics for improving exam timetables', *Annals of Operations Research*, Vol. 218, pp.129–145.

Caramia, M., Dell'Olmo, P. and Italiano, G.F. (2001) 'New algorithms for examination timetabling', *Lecture Notes in Computer Science*, Vol. 1982, pp.230–241.

Carter, M.W., Laporte, G. and Lee, S.Y. (1996) 'Examination timetabling: algorithmic strategies and applications', *Journal of the Operational Research Society*, Vol. 3, pp.373–383.

Casey, S. and Thompson, J. (2003) 'GRASPing the examination scheduling problem', *Lecture Notes in Computer Science*, Vol. 2740, pp.232–244.

Ceder, A. (2011) 'Optimal multi-vehicle type transit timetabling and vehicle scheduling', *Procedia Social and Behavioral Sciences*, Vol. 20, pp.19–30.

Dowsland, K.A. and Thompson, J.M. (2012) 'Simulated annealing', in Rozenberg, G. *et al.* (Eds.): *Handbook of Natural Computing*, Springer, Berlin, pp.1623–1655.

Dun, Y.J., Wang, Q. and Shao, Y.B. (2014) 'A simulated annealing genetic algorithm for solving timetable problems', in Cao, B.Y. and Nasseri, H. (Eds.): *Fuzzy Information and Engineering and Operations Research and Management*, Springer-Verlag, Berlin, pp.365–374.

Gogos, C., Alefragis, P. and Housos, E. (2012) 'An improved multi-staged algorithmic process for the solution of the examination timetabling problem', *Annals of Operations Research*, Vol. 194, pp.203–221.

Haspeslagh, S., De Causmaecker, P., Schaerf, A. and Stolevik , M. (2014) 'First international nurse rostering competition 2010', *Annals of Operations Research*, Vol. 218, pp.221–236.

Jaradat, G., Ayob, M. and Ahmad, Z. (2014) 'On the performance of scatter search for post-enrolment course timetabling problems', *Journal of Combinatorial Optimization*, Vol. 27, pp.417–439.

Kirkpatrick, S., Gellat, C.D. and Vecchi, M.P. (1983) 'Optimisation by simulated annealing', *Science*, Vol. 220, pp.671–680.

Lee, S.Y., Kim, Y.D. and Lee, H.J. (2009) 'Heuristic algorithm for a military training timetabling problem', *Asia Pacific Management Review*, Vol. 14, pp.289–299.

Malaguti, E. and Toth, P. (2010) 'A survey on vertex coloring problems', *International Transactions in Operational Research*, Vol. 17, pp.1–34.

Merlot, L.T.G., Boland, N., Hughes, B.D. and Stuckey, P.J. (2003) 'A hybrid algorithm for the examination timetabling problem', *Lecture Notes in Computer Science*, Vol. 2740, pp.207–231.

MirHassani, S.A. (2006) 'Improving paper spread in examination timetables using integer programming', *Applied Mathematics and Computation*, Vol. 179, pp.702–706.

Monticelli, A.J., Romero, R. and Asada, E.N. (2008) 'Fundamentals of simulated annealing', in Lee, K.Y. et al. (Eds.): *Modern Heuristic Optimization Techniques*, The Institute of Electrical and Electronics Engineers, Inc., Wiley Online Library, pp.123–146.

Müller, T. (2014) 'Real life examination timetabling problems', *Journal of Scheduling*, Vol. 19, pp.257–270.

Nesmachnow, S. (2014) 'An overview of metaheuristics: accurate and efficient methods for optimization', *International Journal of Metaheuristics*, Vol. 3, pp.320–347.

Pais, T.C. and Amaral, P. (2012) 'Managing the tabu list length using a fuzzy inference system: an application to examination timetabling', *Annals of Operations Research*, Vol. 194, pp.341–363.

Petrovic, S. and Burke, E.K. (2004) 'University timetabling', in Leung J. (Ed.): *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, CRC Press, Boca Raton, pp.1–23.

Pillay, N. (2014) 'A review of hyperheuristics for educational timetabling', *Annals of Operations Research*, Vol. 239, pp. 3–38.

Pillay, N., and Banzhaf, W. (2009) 'A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem', *European Journal of Operational Research*, Vol. 197, pp.482–491.

Qu, R. and Burke, E.K. (2009) 'Hybridisations within a graph-based hyper-heuristic framework for university timetabling problems', *Journal of the Operational Research Society*, Vol. 60, pp.1273–1285.

Qu, R., Burke, E.K., McCollum, B., Merlot, L.T.G. and Lee, S.Y. (2009) 'A survey of search methodologies and automated system development for examination timetabling', *Journal of Scheduling*, Vol. 12, pp. 55–89.

Sabar, N.R., Ayob, M., Qu, R. and Kendall, G. (2012) 'A graph coloring constructive hyper-heuristic for examination timetabling problems', *Applied Intelligence*, Vol. 37, pp.1–11.

Suman, B. and Kumar, P. (2006) 'A survey of simulated annealing as a tool for single and multiobjective optimization', *Journal of the Operational Research Society*, Vol. 57, pp.1143–1160.

Thompson, J.M. and Dowsland, K.A. (1995) 'General cooling schedules for a simulated annealing based timetabling system', *Lecture Notes in Computer Science*, Vol. 1153, pp.345–363.

Thompson, J.M. and Dowsland, K.A. (1996) 'Variants of simulated annealing for the examination timetabling problem', *Annals of Operations Research*, Vol. 63, pp.105–128.

Thompson, J.M. and Dowsland, K.A. (1998) 'A robust simulated annealing based examination timetabling system', *Computers and Operations Research*, Vol. 25, pp.637–648.

Woumans, G., De Boeck, L., Beliën, J. and Creemers, S. (2016) 'A column generation approach for solving the examination-timetabling system', *European Journal of Operational Research*, Vol. 253, pp.178–194.