European Journal of Operational Research 203 (2010) 550–558

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

# A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems

Defu Zhang [a,*], Yongkai Liu [a], Rym M'Hallah [b], Stephen C.H. Leung [c]

[a] Dep. of Computer Science, Xiamen University, Xiamen 361005, China
[b] Dep. of Statistics and Operations Research, Kuwait University, P.O. Box 5969, Safat 13060, Kuwait
[c] Department of Management Sciences, City University of Hong Kong, Hong Kong

ARTICLE INFO

*Article history:*
Received 16 July 2008
Accepted 14 September 2009
Available online 19 September 2009

*Keywords:*
Timetabling
Simulated annealing
Extended neighborhood

ABSTRACT

This paper approximately solves the high school timetabling problem using a simulated annealing based algorithm with a newly-designed neighborhood structure. In search for the best neighbor, the heuristic performs a sequence of swaps between pairs of time slots, instead of swapping two assignments as in a standard simulated annealing. The computational results show that the proposed heuristic, which is tested on two sets of benchmark instances, performs better than existing approaches.



## 1. Introduction

Many institutions (academic, health, transportation, sport, etc.) in the world face timetabling problems. For a relatively large institution, this problem is a challenging combinatorial task; thus, efficient timetabling tools should be made available to concerned decision-makers wishing to generate conflict-free flexible timetables.

Timetabling consists in identifying an "optimal" allocation of a given set of events (courses, exams, surgeries, sport events) and resources (teachers, exam proctors, nurses, medical doctors) over space (classrooms, operating rooms, sport fields) and time. For example, high school timetabling consists in allocating courses and teachers to groups of students over time slots (or periods) and assign each pair of (teacher, group of students) during a given period to a classroom. This allocation, which strives to satisfy many objectives, is subject to a set of constraints that are problem dependent. For instance, in high school timetabling, some subjects such as science and mathematics may require two consecutive periods. The science classes require special classrooms. Teachers may be unavailable during some time periods, and some subjects should be taught during the first or last period of the day.

Timetabling constraints can be classified into hard and soft constraints. Any feasible timetable has to satisfy all hard constraints. For instance, the constraint that a teacher cannot be assigned to two classes during the same time period is a hard constraint that has to be satisfied by any feasible schedule. On the other hand, a feasible schedule does not need to satisfy the soft constraints but should strive to maximize their degree of satisfaction. In fact, soft constraints are generally formulated as part of the objective function of the problem. For example, imposing that teachers have a continuous schedule throughout the day is a soft constraint, that is included as part of the objective function.

Timetabling problems faced by academic institutions differ from one institution to another. Most institutions face the timetabling problem in its primary form, which is to allocate sessions and rooms to lectures so as to satisfy a given set of hard and soft constraints. However, each institution has some unique combination of constraints depending on the pertinent policies that usually differ from one institution to another. For example, university timetables are quite different from high school timetables. High school students require constant supervision during the day; thus, should have their schedules continuous. In addition, their teachers generally have much higher teaching loads than their university counterparts. Subsequently, high school timetabling problems may become computationally expensive. Worst, they become more complex as the number of enrolled students (thus, the number of classes) increases. In fact, the larger the problem's size, the more constraints any feasible schedule has to satisfy, and the more conflicting objectives such a schedule has to fulfill. Therefore, the generation of "high-quality" timetables is not a straightforward task.

As pinpointed in the numerous literature reviews [15,22,30,33,34], various timetabling approaches have been investigated. Yet, no specific one can be applied universally due to the specifics of individual problems. The newer approaches focus on bridging the gap between research and the real world applications they

* Corresponding author. Tel.: +86 592 5918207; fax: +86 592 2580035.
*E-mail address:* dfzhang@xmu.edu.cn (D. Zhang).

0377-2217/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2009.09.014

tackle [22,23]. To create better understanding between researchers and practitioners, PATAT and WATT have organized two international timetabling competitions where emerging techniques were tested on real world models. Details relative to these competitions (including benchmark instances, assessment criteria, and proposed approaches) are available at http://www.cs.qub.ac.uk/itc2007 and http://www.idsia.ch/Files/ttcomp2002/. However, the objectives and constraints of these instances are quite different from those of the problem at hand.

Timetabling is generally tackled using mathematical programming, artificial intelligence, or meta-heuristics. Birbas et al. [13] presented a 0–1 integer programming model for the timetabling problem of Greek high schools. In their model, a binary variable indicates whether or not a specific lesson to be taught by a given teacher is to be held at a specific time of the week. The model generates timetables that satisfy all the hard and soft constraining rules of the schools. Daskalaki et al. [24] and Daskalaki and Birbas [25] designed a mixed-integer program for a timetabling problem related to Greek universities. In the latter paper, a two-stage relaxation procedure is applied. In the first stage, the computationally heaviest constraints are initially relaxed. In the second stage, the relaxed constraints are introduced afresh and the timetable is generated on a day-by-day basis to find local optima for the original problem. Papoutsis et al. [32] modeled the high school timetabling problem as a set partitioning problem, and solved it using a column generation approach. Each binary variable (or column) indicates whether or not a feasible weekly schedule of a teacher is included in the timetable. Wood and Whitaker [43] formulated a non-linear goal program to the secondary school timetabling problem, where students freely choose their courses of study from a list of subjects. Mirrazavi et al. [31] developed a two-phase approach for the university timetabling problem. The first phase allocates rooms to classes whereas the second phase assigns a time period to each class. Both phases are modeled as integer goal programs. The approach applies a preprocessing module to remove redundant solutions prior to initiating the optimization step. Asratian and de Werra [9] and de Werra et al. [42] considered a generalized class-teacher model that extends the basic "class-teacher model" of timetabling to situations which occur, for example, in the basic training programs of universities and schools, and designed a network flow algorithm for the solution. Al-Yakoob and Sherali [6,7] designed mixed-integer programming models and algorithms that assign sections of male, female and joint classes to time periods.

Solving large sized instances of timetabling problems using integer or constraint programming [27] remains a challenge. This is not surprising since timetabling problems are NP-complete. As an alternative, researchers have focused on solving this problem using neural networks [19,20] and meta-heuristics such as ant colony optimization [26,36], genetic algorithms [18,29,40], tabu search [1,8,11], evolutionary search [12], simulated annealing (SA) [2,4], programmed search [27], and hyper heuristics [17,21]. Among these approximate approaches, SA has proven quite effective in tackling timetabling related problems such as school, high school, and exam timetabling [2,4,15,16]. For example, Bullnheimer [14] developed a model suitable for small scale exam timetabling problems. He formulated a quadratic assignment problem, transformed it into quadratic semi assignment problems, and used SA to demonstrate the model's ability to generate schedules that satisfy students. Johnson [28] incorporated SA into his cluster approach for exam timetabling. Thompson and Dowsland [38,39] investigated the robustness of an SA approach to the exam timetabling problem and undertook a sensitivity analysis of the solution to the adaptive cooling schedule and to the choice of neighborhood. Abramson et al. [4], which considered high school timetabling, applied an SA with a newly-designed special dynamic cooling schedule. The temperature is increased or decreased adaptively, according to the number of successful moves. Finally, Avella et al. [10] incorporated SA into the first phase of their algorithm with the objective of generating an initial feasible timetable.

This paper adapts SA to the high school timetabling problem and designs a new neighborhood structure that yields high-quality solutions within a reduced runtime. The application of the new algorithm to one typical real-world data set and one randomly generated synthetic data set of high school timetabling problems (i.e., hdtt and g★tt where ★ = b or p) shows its efficiency.

This paper is organized as follows. Section 2 defines the high school timetabling problem, and models it as a 0–1 integer program. Section 3 presents the basics of SA, explains the new neighborhood structure, and details the proposed algorithm. Section 4 assesses and compares the performance of the new algorithm to that of existing approaches. Finally, Section 5 provides a summary and future extensions.

## 2. Problem definition

The high school timetabling problem involves assigning teachers to classes (or groups of students). Each class is assigned a fixed classroom where the class meets for all its lectures, whereas teachers rotate among classes; thus, among classrooms. The assignment of teachers to classes has to satisfy a set of hard and soft constraints. The set of hard constraints follows:

- **H1** A teacher cannot be assigned to more than one class during any time slot.
- **H2** A class cannot be assigned more than one teacher for any time slot.
- **H3** A teacher $t$, teaching subject (or lesson) $s$, is to be assigned $h_{t,s}$ lectures (or time periods or time slots) per week.
- **H3′** A teacher $t$, teaching subject $s$, and assigned to class $c$ meets $c$ for $h_{c,s}$ lectures per week.
- **H4** A class $c$ must attend $h_c$ lectures per week.
- **H5** The lectures of a teacher are scheduled only when he/she is available.
- **H6** A class should have no free time slot except for the last one during a day.

Let $n_c$ denote the number of classes, $n_s$ the number of subjects, $nt_s$ the number of teachers teaching subject $s$, $s = 1, \ldots, n_s$, $n_p$, the number of periods per working day, and $n_d$, the number of working days per week. In addition, let the binary decision variable $y_{cts} = 1$, $c \in C = \{1, \ldots, n_c\}$, $t \in T_s = \{1, \ldots, nt_s\}$, $s \in S = \{1, \ldots, n_s\}$, if teacher $t$ of subject $s$ is assigned to class $c$. Furthermore, let the binary variable $x_{ctspd} = 1$, $c \in C$, $t \in T_s$, $s \in S$, $p \in P = \{1, \ldots, n_p\}$, $d \in D = \{1, \ldots, n_d\}$, if $y_{cts} = 1$ and teacher $t$ is scheduled to meet class $c$ during period $p$ of day $d$. Finally, let $\mathscr{A}_{t,s}$ (resp. $\overline{\mathscr{A}}_{t,s}$) denote the set of periods when teacher $t$ of subject $s$ is available (resp. unwilling) to teach. The hard constraints can therefore, be modeled as follows:

$$\sum_{c \in C} x_{ctspd} \leqslant 1 \quad t \in T_s,\ s \in S,\ p \in P,\ d \in D, \tag{H1}$$

$$\sum_{c \in C} \sum_{d \in D} \sum_{p \in P} x_{ctspd} = h_{t,s} \quad s \in S,\ t \in T_s, \tag{H2}$$

$$\sum_{s \in S} \sum_{t \in T_s} x_{ctspd} \leqslant 1 \quad c \in C,\ p \in P,\ d \in D, \tag{H3}$$

$$\sum_{d \in D} \sum_{p \in P} x_{ctspd} \geqslant h_{c,s} y_{cts} \quad c \in C,\ t \in T_s,\ s \in S, \tag{H3′}$$

$$\sum_{s\in S}\sum_{t\in T_s}\sum_{d\in D}\sum_{p\in P} x_{ctspd} = h_c \quad c\in C, \tag{H4}$$

$$\sum_{c\in C} x_{ctspd} = 0 \quad \text{if } (p,d)\notin \mathscr{A}_{t,s},\ t\in T_s,\ s\in S, \tag{H5}$$

$$\sum_{s\in S}\sum_{t\in T_s} x_{cts(p+1)d} - \sum_{s\in S}\sum_{t\in T_s} x_{ctspd} \geqslant 0 \quad c\in C,\ d\in D, \tag{H6}$$

$$p = 1,\ldots,n_p-2 \tag{H7}$$

$$\sum_{t\in T_s} y_{cts} = 1 \quad c\in C,\ s\in S, \tag{H8}$$

$$y_{cts} \geqslant x_{ctspd} \quad c\in C,\ s\in S,\ p\in P,\ d\in D,\ t\in T_s. \tag{H7'}$$

Constraints (H8) and (H7′) are added to the mathematical model to ensure that only one teacher is assigned to a subject for a given class, and that only this specific teacher is assigned to the $h_{c,s}$ lectures of class $c$ for subject $s$. Evidently, there are the binary constraints which have not been explicitly stated.

There are some cases in the literature [4,10,11,32,37,41] where the teachers are pre-assigned to teach a given subject to a class. The above model can be easily transformed to account for those cases by setting the appropriate binary variables to their corresponding preset values; i.e., by presetting $y_{cts}$ to 0 and 1, and removing (H8) from the model. Furthermore, when the concept of a subject is completely irrelevant, for simplification purposes or because of the actual problem in its origin, the model can be changed as follows: the index $s$ is dropped from the variables, the set of teachers is defined as $\mathscr{T} = \bigcup_{s\in S} T_s$, the condition $s\in S, t\in T_s$ is replaced by $t\in\mathscr{T}$, and $\sum_{s\in S}\sum_{t\in T_s}$ is substituted by $\sum_{t\in\mathscr{T}}$.

The soft constraints are used as an evaluation tool of the quality of a feasible timetable. They are included into the objective function $z$, which is problem dependent. For instance, **for hdtt problems**, $z$ is the weighted sum of the total penalty (or inconvenience) incurred by the teachers:

$$z = \sum_{s\in S}\sum_{t\in T_s} \gamma_{t,s} z_{t,s},$$

where a large $\gamma_{t,s}$ emphasizes the importance of the timetable of teacher $t$ of subject $s$, denoted hereafter as $(s,t)$. In turn, $z_{t,s}$, the total penalty incurred by teacher $(s,t)$ for a specific timetable, is a weighted sum of six penalties:

$$z_{t,s} = \sum_{i=1}^{6} \theta_i z^i_{t,s},$$

with $z^i_{t,s}$ being the penalty for criterion $i$, and $\theta_i$ its relative importance.

The first penalty $z^1_{t,s}$ indicates the number of times teacher $(s,t)$ is assigned to teach when he/she is unwilling to lecture. That is,

$$z^1_{t,s} = \sum_{(p,d)\in\overline{\mathscr{A}}_{t,s}}\sum_{c\in C} x_{ctspd}.$$

The second penalty $z^2_{t,s}$ tallies the idle time of teacher $(s,t)$ over $D$, where the idle time for a given day $d\in D$ is only measured over the interval $[p'_d, p''_d]$ with $p'_d$ and $p''_d$ being the first and last teaching period of teacher $(s,t)$ on day $d$. That is, the idle time for $d$ is the difference between the range $p''_d - p'_d$ and the number of teaching hours $\sum_{p\in P}\sum_{c\in C} x_{ctspd}$. Subsequently,

$$z^2_{t,s} = \sum_{d\in D}\left[p''_d - p'_d - \sum_{p\in P}\sum_{c\in C} x_{ctspd}\right].$$

The third penalty $z^3_{t,s}$ specifies the number of times teacher $(s,t)$ meets a class $c$ more than once a day but during two non-consecutive periods. Let $\delta_{ctsd} = 1$ if $\sum_{p\in P} x_{ctspd} > y_{cts}$ and for $p = 1,\ldots, n_p-1$, $x_{ctspd} \neq x_{cts(p+1)d}$. It follows that

$$z^3_{t,s} = \sum_{d\in D}\sum_{c\in C}\delta_{ctsd}.$$

The fourth penalty reflects the number of days, over $D$, when the number of contact hours of teacher $(s,t)$ with class $c$ differs from $\frac{1}{n_d}h_{c,s}$, the average daily contact hours for subject $s$. Let $\delta'_{ctsd} = 1$ if $|\sum_{p\in P} x_{ctspd} - \frac{1}{n_d}h_{c,s}y_{cts}| > 1$, and 0 otherwise. Then,

$$z^4_{t,s} = \sum_{c\in C}\sum_{d\in D}\delta'_{ctsd}.$$

The fifth penalty $z^5_{t,s}$ indicates the number of days, over $D$, when the teaching load of teacher $(s,t)$ differs from his/her daily average load $\frac{1}{n_d}h_{t,s}$. Let $\delta''_{tsd} = 1$ if $|\sum_{c\in C}\sum_{p\in P} x_{ctspd} - \frac{1}{n_d}h_{t,s}| > 1$, and 0 otherwise. Then,

$$z^5_{t,s} = \sum_{d\in D}\delta''_{tsd}.$$

The sixth penalty $z^6_{t,s}$ shows the number of times teacher $(s,t)$ is assigned to teach during the last period of any day $d\in D$. That is,

$$z^6_{t,s} = \sum_{d\in D}\sum_{c\in C} x_{ctsn_p d}.$$

For **g★tt instances**, $z$ reflects the degree of non-satisfaction of three soft constraints S1–S3:

- **S1** The teaching load of a teacher should be balanced throughout the week.
- **S2** A class should not be assigned the same subject more than once during a day $d\in D$.
- **S3** The lectures of a teacher should be consecutive with no free time slots.

Specifically, the objective function is driven by six penalties: $z_1,\ldots,z_6$. $z_1$ and $z_2$ are relative to S1; $z_3$ and $z_4$ to S2; and $z_5$ and $z_6$ to S3. $z_1$ is the number of teachers whose teaching hours are not uniformly distributed among the days of the timetable, and $z_2$ is the total number of days (over all teachers over a week) that this uneven distribution of lectures occurs. $z_3$ is the number of classes which are taught the same subject twice or more during a day while $z_4$ tallies (over all classes and days of the week) the number of times this situation occurs. Finally, $z_5$ is the distinct number of teachers whose timetables have idle time slots while $z_6$ is the total number of idle time slots for all teachers over a week.

## 3. Solution approach

The proposed solution approach has two phases with each phase using an SA heuristic with a new extended neighborhood structure. SA is a variant of local search. It is a meta-strategy for optimization by local improvements [5]. It was developed by analogy to the annealing process studied in mechanical statistics. It is a stochastic steepest descent where moves to non-improving neighboring solutions are allowed with a given probability that decreases as the search progresses. These moves are undertaken in hope of escaping local minima. This section explains how SA has been adapted to the high school timetabling at hand, then details the proposed solution approach.

### 3.1. Adapting SA to the high school timetabling problem

Applying SA requires defining a solution configuration, identifying the neighborhood of the current solution, and setting SA's parameters. For the high school timetabling, a **solution** $\tau$ is represented by a two-dimensional matrix of dimensions $n_c \times (n_p n_d)$, with each cell $(c,p')$ indicating the teacher $(s,t)$ assigned to class

**Table 1**
Teachers, their respective subjects, and the corresponding (subject, teacher) code.

| $s$ | Subject | Teacher | $(s,t)$ |
|---|---|---|---|
| 1 | Math | Mary | (1,1) |
| 1 | Math | Monica | (1,2) |
| 2 | Science | Steve | (2,1) |
| 2 | Science | Sam | (2,2) |
| 3 | Literature | Larry | (3,1) |
| 3 | Literature | Laura | (3,2) |
| 4 | History | Henry | (4,1) |

**Table 2**
A timetable $\tau$ for a day $d$ where each cell represents $(s,t)$.

| Class | Period | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| 1 | | | (2,1) | (1,1) | (3,1) |
| 2 | (3,1) | (1,2) | | (2,2) | |
| 3 | (3,2) | (1,1) | (2,2) | (4,1) | (3,2) |
| 4 | (3,2) | (4,1) | | (1,1) | (2,1) |
| 5 | (3,1) | (2,1) | | | (1,2) |

$c$ during period $p$ of day $d$ or is empty if class $c$ is idle during that period with $p' = p + (d-1)n_p$.

For example, let $n_p = n_c = 5, n_s = 4$ (with $s = 1$ for math, $s = 2$ for science, $s = 3$ for literature, and $s = 4$ for history), $nt_1 = nt_2 = nt_3 = 2$, and $nt_4 = 1$. In addition, let the list of teachers and their corresponding subjects be given by Table 1. A timetable $\tau$ for a given day $d \in D$ can be represented by the two-dimensional sub-matrix of Table 2, where the rows correspond to the five classes, and the columns to the five time periods of day $d$. Class 2 is assigned to teacher 1 of subject 3 (i.e., Larry) during the first time period ($p = 1$), to teacher 2 of subject 1 (i.e., Monica) for $p = 2$, and to teacher 2 of subject 2 (i.e., Sam) for $p = 4$. This class is free during the third and fifth periods. This timetable is obviously infeasible. It violates the hard constraints (H1). Indeed, teacher 1 of subject 3 is assigned to classes 2 and 5 for period 1. Similarly, teacher 2 of subject 3 is assigned to classes 3 and 4 for period 1. Thus, the schedules of these two teachers violate (H1). In addition, classes 2, 4, and 5 are free during the third period of $d$; thus, their schedules violate (H7). Evidently, $\tau$ violates many soft constraints. For instance, class 3 has two non-consecutive lectures of subject 3. Most teachers have idle time in their schedules; etc.

A **neighbor** $\tau_N$ of $\tau$ is constructed by swapping the assignments of time slots in $\tau$. For instance, swapping the assignments of $p = 1$ and $p = 4$ for class 4 is equivalent to swapping the lecture times of subjects 3 and 1. It yields the timetable $\tau_N$ displayed in Table 3. Obviously, $\tau_N$ violates less hard constraints than $\tau$ : teacher (3,2) no longer has a conflict during $p = 1$ and teacher (1,1) is conflict free during $p = 4$. However, $\tau_N$ remains infeasible. $\tau_N$ can be further improved if the same swap is applied successively to all $c \in C$; that is, if all lectures (or idle times) scheduled during $p = 1$ are swapped with those scheduled during $p = 4$.

Accepting or rejecting a swap proceeds as follows. When $\tau$ is **infeasible**, a swap is adopted if it results in fewer non-satisfied hard constraints; i.e., if $z^h(\tau_N) < z^h(\tau)$, where $z^h(\tau)$ (resp. $z^h(\tau_N)$) is the number of hard constraints violated by $\tau$ (resp. $\tau_N$). The swap is also accepted with a probability $P_a = \exp^{-\frac{\Delta'}{T}}$ when $\Delta' = z^h(\tau_N) - z^h(\tau) > 0$, where $\Delta'$ is the number of additional non-satisfied hard constraints, and $T$ is the current temperature of the annealing process. On the other hand, when $\tau$ is **feasible**, a swap is adopted if it satisfies all hard constraints and improves or maintains the objective function value; i.e., if $z^h(\tau_N) = 0$, and $z(\tau) \geqslant z(\tau_N)$, where $z(\tau)$ and $z(\tau_N)$ denote the objective function value of $\tau$ and $\tau_N$, respectively. The swap is also accepted with a probability $P_a = \exp^{-\frac{\Delta}{T}}$ if $\tau_N$ satisfies all hard constraints but deteriorates the objective function value; i.e., if $z^h(\tau_N) = 0$, and $\Delta = z(\tau_N) - z(\tau) > 0$, where $\Delta$ is the deterioration of the objective function value.

The swaps are undertaken consecutively. Every time a swap is accepted, the current solution $\tau$ is updated: it is set to $\tau_N$. On the other hand, when a swap is rejected, the current solution is not updated, and a new neighbor is generated. Applying the swaps to a random order of the classes avoids premature convergence of the algorithm.

Consider, for instance, the random permutation of the five classes as 1–3–5–2–4, and let the current solution for a day $d$ be the timetable given by Table 2. Then, the first neighbor is obtained by swapping the assignment of $p = 1$ and $p = 4$ for class 1; that is, swapping (1,1) with the free time slot. This swap reduces the number of violations of the hard constraints; thus, the current solution is set to this neighbor. The following neighbor is the result of the swap of the assignment of $p = 1$ and $p = 4$ for class 3: it has (4,1) swapped with (3,2). Again, this neighbor reduces the number of violations of the hard constraints; thus, it is retained as the current solution. The next neighbor swaps the assignment (3,1) with a free time slot for class 5. Since it reduces the non-feasibility of the timetable, this neighbor is retained as the current solution. The next two neighbors obtained by swapping (2,2) and (3,1) for class 2 and (1,1) with (3,2) for class 4 are non-improving neighbors since they do not reduce the number of non-satisfied constraints. In this case, they are rejected, and the resulting current solution is given by Table 4.

The temperature, the length $L$ of the plateau, and the stopping criterion are the **parameters** of SA. The temperature is initially set at a level $T_0$ that guarantees a fixed initial probability of acceptance (eg., 0.95 or 0.9). It is adjusted every $L$ iterations. If $T_k$ is the temperature at plateau $k$, then the temperature at the next plateau is $T_{k+1} = \alpha T_k$, where the cooling rate $\alpha \in [0,1]$; for example, $\alpha = 0.93$. This cooling strategy makes the acceptance of non-improving neighbors less likely as the search progresses. The algorithm stops when the temperature reaches a pre-specified threshold level.

### 3.2. Detailed algorithm of the proposed approach

Specifically, the proposed approach, whose pseudocode is detailed in Table 5, is a two-phase heuristic. The first phase con-

**Table 3**
A neighbor of $\tau$ obtained by swapping the assignments of $p = 1$ and $p = 4$ for $c = 4$.

| Class | Period | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| 1 | | | (2,1) | (1,1) | (3,1) |
| 2 | (3,1) | (1,2) | | (2,2) | |
| 3 | (3,2) | (1,1) | (2,2) | (4,1) | (3,2) |
| 4 | (1,1) | (4,1) | | (3,2) | (2,1) |
| 5 | (3,1) | (2,1) | | | (1,2) |

**Table 4**
A neighbor of $\tau$ obtained using the extended neighborhood structure.

| Class | Period | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| 1 | (1,1) | | (2,1) | | (3,1) |
| 2 | (3,1) | (1,2) | | (2,2) | |
| 3 | (4,1) | (1,1) | (2,2) | (3,2) | (3,2) |
| 4 | (3,2) | (4,1) | | (1,1) | (2,1) |
| 5 | | (2,1) | | (3,1) | (1,2) |

**Table 5**
Algorithm of the proposed approach.

**Phase one**: Identifying a feasible timetable $\tau$
**Initialization Step**
Randomly generate a timetable $\tau$ that satisfies (H2), (H3′) and (H4)
Compute $z^h(\tau)$ the number of hard constraints violated by $\tau$
Set $k = 1$ and $T_k = T_0^1$, where $k$ and $T_k$ are the number and temperature of the current plateau, and $T_0^1$ is a preset initial temperature for phase one
Iterative Step
**While**($z^h(\tau)>0$ and $T_k>T^1$) // $T^1$: threshold temperature for phase one
**Do for** $\ell = 1$–$L_1$ // $L_1$: length of the plateau for phase one
- Generate a random permutation $\Pi_C$ of $C$
- Choose a period $p_1 \in P$ such that the assignments during $p_1$, for any day $d \in D$, violate one or more of the hard constraints
- Choose, randomly, a period $p_2 \in P \setminus \{p_1\}$

**Do for** $i = 1$–$n_c$
– Swap the assignment for $p_1$ and $p_2$ for the $i$th class in $\Pi_C$, and let $\tau_N$ be the resulting neighbor
– Evaluate $z^h(\tau_N)$, and compute $\Delta' = z^h(\tau_N) - z^h(\tau)$
– If $\Delta' \leq 0$, set $z^h(\tau) = z^h(\tau_N)$
– Else, generate a random number $r$ from the continuous Uniform[0,1]; If $r < \exp^{-\frac{\Delta'}{T_k}}$, set $z^h(\tau) = z^h(\tau_N)$
**End Do**
**End Do**
Set $T_{k+1} = \alpha_1 T_k$, and $k = k + 1$. // $\alpha_1$: cooling rate for phase one
**End While**
**Phase Two**: Identifying a (near) optimal timetable
**Initialization Step**
Compute $z(\tau)$ the value of the objective function at $\tau$
Set $k = 1$ and $T_k = T_0^2$, where $T_0^2$ is a preset initial temperature for phase two
**Iterative Step**
**While**($T_k>T^2$) // $T^2$: threshold temperature for phase two
**Do for** $\ell = 1$–$L_2$ // $L_2$: length of the plateau for phase two
Generate a random permutation $\Pi_P$ of $P$
Set $p_1$ as the first item in $\Pi_P$, and remove it from $\Pi_P$
**While** ($|\Pi_P| \geqslant 1$)
**Do for** $j = 1$–$|\Pi_P|$
- Set $p_2$ to the $j$th item of $\Pi_P$
- Generate a random permutation $\Pi_C$ of $C$

**Do for** $i = 1$–$n_c$
– Swap the assignment for $p_1$ and $p_2$ for the $i$th class in $\Pi_C$, and let $\tau_N$ be the resulting neighbor
– If $z^h(\tau_N) > 0$, goto **End Do**
– Evaluate $z(\tau_N)$, and compute $\Delta = z(\tau_N) - z(\tau)$
– If $\Delta \leqslant 0$, set $z(\tau) = z(\tau_N)$
– Else, generate a random number $r$ from the continuous Uniform[0,1]; if $r < \exp^{-\frac{\Delta'}{T_k}}$, set $z(\tau) = z(\tau_N)$
**End Do**
**End Do**
**End While**
**End Do**
Set $T_{k+1} = \alpha_2 T_k$, and $k = k + 1$. // $\alpha_2$: cooling rate for phase two
**End While**

structs an initial feasible solution whereas the second phase – which starts from the initial feasible solution identified in phase one – searches for a near optimum that minimizes the degree of non-satisfaction of the soft constraints.

**Phase one** starts with an initialization step, which chooses an initial solution, sets the current plateau, and specifies its temperature. Specifically, it randomly generates a timetable $\tau$ that satisfies (H2), (H3′) and (H4). That is, $\tau$ ensures the assignment of a teacher $t \in T_s$ for every subject $s \in S$ for every class $c \in C$, that at most a single lecture is assigned to each class for each time period, and that each class $c \in C$ gets $h_{c,s}$ lectures for each subject $s \in S$. In case, the teachers are pre-assigned to the classes, then phase one generates a timetable that respects this pre-assignment. Next, it sets the current plateau $k = 1$, and the current temperature $T_k$ to $T_0^1$.

Phase one proceeds, then, with an iterative phase (corresponding to the While loop of Phase one of Table 5) that moves the search from one plateau to the next. For plateau $k$, it investigates $L_1$ neighborhoods of $\tau$ updating $\tau$ every time either an improving neighbor or a neighbor with an improvement potential is identified. Specifically, it sets $\ell$, the counter of the number of neighborhoods investigated at the current plateau $k$ to 1. It then obtains a random permutation $\Pi_C$ of $C$, and proceeds by generating successively the neighbors of $\tau$. It chooses a period $p_1 \in P$ such that the assignments, in $\tau$, of the pairs $(s,t)$ to classes during $p_1$, for any day $d \in D$, violate one or more of the hard constraints. Next, it chooses randomly another period $p_2 \in P \setminus \{p_1\}$, and swaps, successively, for $i = 1$–$n_c$, the assignments corresponding to $p_1$ and $p_2$ for the $i$th class in $\Pi_C$. Each swap results in a new neighbor $\tau_N$ of $\tau$. If $z^h(\tau_N) \leqslant z^h(\tau)$, the heuristic accepts the swap. Otherwise, it generates a random number $r$ from the continuous Uniform[0,1], and compares it to $\exp^{-\frac{\Delta'}{T_k}}$, where $\Delta' = z^h(\tau_N) - z^h(\tau)$. The heuristic accepts the swap if $r < \exp^{-\frac{\Delta'}{T_k}}$, and rejects it otherwise. When the heuristic accepts the swap, it updates the current timetable by setting $\tau = \tau_N$. Once it has performed all the $n_c$ swaps and either accepted or rejected each of them, the heuristic increases the counter $\ell$.

If $\ell \leqslant L_1$ while $\tau$ is infeasible, the heuristic proceeds by generating a new permutation $\Pi_C$ of the classes and choosing two periods $p_1$ and $p_2$ as described above. On the other hand, when $\ell$ exceeds $L_1$, the heuristic moves to a new plateau; signaling the end of the first do loop of phase one. Subsequently, it cools the temperature setting $T_{k+1} = \alpha_1 T_k$, increments $k$ by one, and initializes $\ell$ to 1. It then repeats the above steps for the new plateau.

The heuristic keeps moving from plateau to plateau as long as $\tau$ is infeasible and the temperature $T_k > T^1$, a preset threshold temperature. If the heuristic stops with $\tau$ being infeasible, it generates a new initial timetable and reapplies all the above steps. (However, this never occurred during the computational investigation.) Phase

one stops at any of the above steps when it identifies a feasible timetable $\tau$; i.e., when $z^h(\tau) = 0$.

**Phase two** starts its search with the feasible timetable identified in phase one, and strives to improve the value of $z$ while maintaining feasibility. The initialization step sets the current plateau $k = 1$, and the current temperature $T_k$ to $T_0^2$.

Phase two proceeds, with an iterative phase (corresponding to the While loop of phase two of Table 5) that moves the search from one plateau to the next. For plateau $k$, it investigates $L_2$ neighborhoods of $\tau$ as follows. First, it initializes $\ell = 1$. Then, it generates a random permutation $\Pi_P$ of $P$. It sets $p_1$ as the first period appearing in $\Pi_P$, removes it from $\Pi_P$ setting $\Pi_P = \Pi_P \setminus \{p_1\}$, then selects randomly a period $p_2$ from $\Pi_P$. Subsequently, it considers a random permutation $\Pi_C$ of $C$, and generates successively the neighbors of $\tau$ by swapping the assignment of $p_1$ and $p_2$ for every class $c \in C$, with the classes considered according to their order in $\Pi_C$. A neighbor $\tau_N$ of $\tau$ is automatically rejected if it violates any hard constraint (i.e., $z^h(\tau_N) > 0$). On the other hand, when $\tau_N$ is feasible, the heuristic accepts the swap if $z(\tau_N) \leqslant z(\tau)$, or if $z(\tau_N) > z(\tau)$ but a randomly generated number $r \in [0, 1]$ is less than $\exp^{-\frac{\Delta}{T_k}}$ where $\Delta = z(\tau_N) - z(\tau)$. When the heuristic accepts the swap, it updates the current timetable setting $\tau = \tau_N$. Once it has performed all the $n_c$ swaps and either accepted or rejected each of them, the heuristic restarts this loop by setting $p_1$ to the new first period in $\Pi_P$, removes it from $\Pi_P$, and selects randomly a period $p_2$. It repeats this step as long as $|\Pi_P|$, the number of elements in $\Pi_P$, is greater than one. When $|\Pi_P| = 1$, the heuristic increases the counter $\ell$ setting it equal to $\ell + 1$.

When $\ell$ exceeds $L_2$, the heuristic moves to a new plateau; signaling the end of the first do loop of phase two. Subsequently, it cools the temperature setting $T_{k+1} = \alpha T_k$, increments $k$ by one, and initializes $\ell$ to 1. It then applies the above steps for the new plateau. The heuristic keeps moving from plateau to plateau as long as the temperature $T_k > T^2$.

## 4. Computational results

The purpose of the computational investigation is to compare the performance of the proposed approach to existing algorithms. The proposed approach is coded in `C++`, and is run on a `Pentium IV`, 2.60 GHz and 512 MB of RAM, under the Linux environment. It is tested on two sets of benchmark instances: hdtt and g★tt.

### 4.1. First benchmark set

The set hdtt has five instances: hdtt4–hdtt8. It was originally proposed by Abramson and Dang [3], and is publicly available from the OR-Library (http://www.ms.ic.ac.uk). Even though small-sized in terms of the total number of teachers and classes, each of these instances has a very dense timetable that uses **all** available time periods. Differently stated, each of these instances has a feasible solution; however, identifying it is extremely hard [37]. Indeed, the "hd" in the label of the instances stands for "hard". Thus, these instances are used, as in [37], to assess the proposed approach as an optimization tool.

An instance hdtt$m$, $m = 4, \ldots, 8$, is characterized by $n_c = m, \sum_{s \in S} nt_s = m$, and $n_d n_p = 30$. It is run with the parameters of the proposed approach set as follows: $T_0^1 = 4$, $T_0^2 = 1$, $T^1 = 1$, $T^2 = .01$, $\alpha_1 = \alpha_2 = .90$, $L_1 = 20$, and $L_2 = 2$. In addition, $\overline{\mathscr{A}}_{\ell,\phi}$ is set to the empty set for all $s \in S$ and $t \in T_s$; that is, teachers are willing to teach during any of the $n_d n_p$ periods. Moreover, $\gamma_{t,s} = 1$, for all $s \in S$ and $t \in T_s$, and $\theta_2 = \ldots, \theta_6 = 1$ whereas $\theta_1 = 0$. The aforementioned parameter levels and restrictions are imposed to allow for the comparison of the results to their counterparts in the literature. Each instance is run 20 times. The best solution $z^*$, the average solution $\bar{z}$, and the average run time $\overline{\text{CPU}}$ (in s) over the 20 runs are compared to their counterparts in the literature obtained via the simulated annealing heuristic SA2 of [35], the neural network algorithm NN-TT3 of [37], and the local search heuristic SA-VLSN of [10]. SA2 and NN-TT3 are reported to be the best approaches among those tested in [37]. SA-VLSN performs better than SA2 and NN-TT3, and is more recent.

SA2 is a meta-heuristic based solver. It encodes the solutions using a linked list representation of the problem, rather than binary matrices. The lists and their lengths are dynamic. They are subject to several transition operators: swap, move, reposition, inversion, add, drop, and change. The probability that an operator is applied is adaptive: The better the solutions it yields, the more frequently it is applied and the more its transition probability is raised. SA2 applies a cooling schedule which reheats the temperature every 10,000 solutions. It stops when it reaches a zero cost or 2750 second of runtime.

NN-TT3 is a modified discrete Hopfield neural network. Its modifications enable the neural network to work in short bursts of gradient descent, followed by random but controlled perturbations that allow its escape from local minima. Furthermore, the modifications guarantee a dynamic number of iterations at each descent, and maintain the run time reasonable.

SA-VLSN combines a standard SA with a very large-scale neighborhood search (VLSN). In fact, it applies VLSN to the best solution obtained by the standard SA. VLSN consists in solving iteratively a mixed-integer programming (MIP) problem with all teachers' schedules prefixed except for two. If it improves the current solution, the resulting neighbor is adopted, the list $\mathscr{T}$ of teachers is reset to $\bigcup_{s \in S} T_s$, and the MIP problem is solved again using Cplex (evoked via ILOG); otherwise the two teachers are removed from $\mathscr{T}$, two other teachers are chosen randomly, and the problem is resolved. VLSN is repeated until $\mathscr{T} = \emptyset$; that is, until there is no improving solution in the extended neighborhood of the current solution. The results highlight the need for the combination of SA with an extended neighborhood search since applying VLSN from an arbitrary feasible solution slows its convergence, and does not necessarily yield good quality solutions. In addition, the value of any solution obtained by the standard SA is evidently greater than or equal to the value of SA-VLSN's solution.

Table 6 summarizes the results of the comparison of the proposed approach to SA2, NN-TT3, and SA-VLSN. Column 1 indicates the instance. Columns 2–4, 5–7, 8–10, and 11–13 report $z^*$, $\bar{z}$, and $\overline{\text{CPU}}$ for the proposed approach, NN-TT3, SA2, and SA-VLSN, respectively. SA2 and NN-TT3 were run on a `Pentium II`, 333 MHz and 128 MB of RAM whereas SA-VLSN was run for the hdtt instances on a `Pentium II`, 400 MHz and 256 MB of RAM, a machine equivalent to that used in [37] (according to [10]).

Even though the four approaches reach the optimum during one of the 20 runs, the proposed approach performs consistently better than SA2 or NN-TT3, and slightly better than SA-VLSN. In fact, it reaches the optimum on every run for instances hdtt4–hdtt7; a performance that is not matched by SA2, NN-TT3, or SA-VLSN. In addition, its $\bar{z}$ for hdtt8 is less than its counterparts: 1.2 for NN-TT3, 1.9 for SA2, and 0.6 for SA-VLSN. Finally, the runtime of the proposed approach is very small in comparison to that of SA2 and NN-TT3. It is true that machine differences account for some of this gap, but it can be inferred that the proposed approach is faster than either SA2 or NN-TT3. Subsequently, it is more efficient than SA2 (with its complex reheating and cooling schedules) and NN-TT3 (with its modified dynamics which are meant to yield faster and better results).

To further substantiate the above inference regarding runtime and to compare the runtime of the proposed approach and of SA-VLSN, we ran the code of SA-VLSN (publicly available at http://acheron.icc.ru/igor/hdtt.cpp) and our code on our machine. We added

**Table 6**
Comparative results for the hdtt instances.

| Instance | Proposed approach | | | NN-TT3 | | | SA2 | | | SA-VLSN | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $z^*$ | $\bar{z}$ | $\overline{CPU}$ | $z^*$ | $\bar{z}$ | $\overline{CPU}$ | $z^*$ | $\bar{z}$ | $\overline{CPU}$ | $z^*$ | $\bar{z}$ | $\overline{CPU}$ |
| hdtt4 | 0 | 0 | 0.20 | 0 | 0.5 | 109.2 | 0 | 0 | 14.57 | 0 | 0 | 0.63 |
| hdtt5 | 0 | 0 | 0.58 | 0 | 0.5 | 146.3 | 0 | 0.3 | 41.20 | 0 | 0 | 1.54 |
| hdtt6 | 0 | 0 | 3.65 | 0 | 0.7 | 226.9 | 0 | 0.8 | 123.02 | 0 | 0 | 3.47 |
| hdtt7 | 0 | 0 | 5.36 | 0 | 1.0 | 323.7 | 0 | 1.2 | 256.59 | 0 | 0.1 | 7.29 |
| hdtt8 | 0 | 0.4 | 9.14 | 0 | 1.2 | 431.3 | 0 | 1.9 | 471.20 | 0 | 0.6 | 14.68 |

**Table 7**
Comparative results of the proposed approach and SA-VLSN for the hdtt instances.

| Instance | Proposed approach | | | SA-VLSN | | |
|---|---|---|---|---|---|---|
| | $z^*$ | $\bar{z}$ | $\overline{CPU}$ | $z^*$ | $\bar{z}$ | $\overline{CPU}$ |
| hdtt4 | 0 | 0 | 0.0086 | 0 | 0 | 0.3386 |
| hdtt5 | 0 | 0 | 0.0156 | 0 | 0 | 0.6180 |
| hdtt6 | 0 | 0 | 0.0320 | 0 | 0.1 | 1.3914 |
| hdtt7 | 0 | 0 | 0.2281 | 0 | 0.5 | 2.5710 |
| hdtt8 | 0 | 0 | 0.5501 | 0 | 4.0 | 5.6914 |

to the former a random number generator and a run time counter. In addition, we modified our code to make it use the same random number generator. Finally, we initiated both codes with the same seed; subsequently, with the same randomly generated solutions. Table 7 displays the resulting best solution values, the average solution values and the average run time over 20 runs. It shows that the proposed approach obtains consistently a zero value for the objective function within a much smaller time than SA-VLSN. Differently stated, the proposed approach with its simple new neighborhood structure provides results that are similar to or better than those of SA-VLSN within a very reduced time.

### 4.2. Second benchmark set

The second benchmark set, g★tt, has 11 instances: gbtt1–gbtt7 [12], and gptt1–gptt4 [32,41]. Even though a description of instances gbtt1–gbtt6 is available through the web site http://prlab.ceid.upatras.gr/timetabling/, gbtt6 could not be tested due to the lack of some important data. The g★tt set corresponds to real-life problems emanating from Greek high schools settled in the city of Patras. It reflects problems that are less dense than those of the first set, but are larger in size with more teachers, classes, subjects, and time periods. This set is run with the parameters of the proposed approach fixed as follows: $T_0^1 = 1$, $T_0^2 = 1$, $T^1 = .1$, $T^2 = .05$, $\alpha_1 = .95$, $\alpha_2 = .90$, $L_1 = 100$, and $L_2 = 10$. The characteristics of each of these instances are provided in Table 8.

**Table 8**
Characteristics of the gbtt and gptt instances.

| Instance | gbtt | | | | | | gptt | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 7 | 1 | 2 | 3 | 4 |
| $\sum_{s\in S} nt_s$ | 34 | 35 | 19 | 19 | 18 | 35 | 11 | 15 | 17 | 21 |
| $n_c$ | 11 | 11 | 6 | 7 | 6 | 13 | 5 | 6 | 7 | 9 |
| $n_d n_p$ | 35 | 35 | 35 | 35 | 35 | 35 | 30 | 35 | 30 | 35 |

For the gbtt instances, the timetables generated using the proposed approach are compared to those generated by the adaptive evolutionary algorithm (EA) of [12] and to the real-world ones used at the Greek high schools. EA's initial population, whose size equals 25, consists of random timetables constructed iteratively by assigning a class $c \in C$ and period $p \in P$ to a teacher $t \in \bigcup_{s\in S} T_s$ while maintaining the feasibility of H1, H2 and H3. The fitness of each timetable is a weighted function of the violations of the hard and soft constraints with the weights attributed to the violations of the hard constraints being much larger than those for the soft constraints. EA does not apply a crossover operator, and uses a linear ranking selection. Each chromosome is subject, with a probability of 0.05, to two mutation operators: period, and bad period mutation. Both operators preserve the satisfaction of H1, H2 and H3. They swap periods of two teachers assigned to a same class $c$ except that the period mutation chooses the periods randomly while the bad period mutation selects the periods causing the largest cost. EA applies a simple elitism schema. It stops after 10,000 generations.

Table 9 compares the timetables constructed by the proposed approach to those constructed using EA and those used at schools for the gbtt instances. Column 1 indicates the instance number. Columns 2–7, 8–13, and 14–19 display the values of $z_i$, $i = 1, \ldots, 6$, for the timetable obtained by the proposed approach, by EA, and the real-world one, respectively. The values of $z_1$, $z_2$ are relative to the distribution of teachers, of $z_3$, $z_4$ to the distribution of lessons, and of $z_5$, $z_6$ to the gaps in the teachers's schedules.

**Table 10**
Comparing the average run time of the proposed approach and of EA [12].

| gbtt | Proposed approach (second) | | | EA (minute) |
|---|---|---|---|---|
| | Phase 1 | Phase 2 | Total | |
| 1 | 1.23 | 138.63 | 139.86 | 30 |
| 2 | 1.08 | 170.21 | 171.29 | 30 |
| 3 | 0.47 | 47.14 | 47.61 | 24 |
| 4 | 0.59 | 32.66 | 33.25 | 24 |
| 5 | 0.56 | 53.61 | 54.17 | 22 |
| 7 | 6.48 | 202.61 | 209.09 | 45 |

**Table 9**
Comparing timetables constructed by the proposed approach with those constructed using EA [12] and those used at schools for the gbtt instances.

| gbtt | Proposed approach | | | | | | EA | | | | | | Used in high schools | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $z_1$ | $z_2$ | $z_3$ | $z_4$ | $z_5$ | $z_6$ | $z_1$ | $z_2$ | $z_3$ | $z_4$ | $z_5$ | $z_6$ | $z_1$ | $z_2$ | $z_3$ | $z_4$ | $z_5$ | $z_6$ |
| 1 | 6 | 11 | 1 | 1 | 11 | 17 | 18 | 40 | 1 | 1 | 25 | 29 | 21 | 48 | 1 | 1 | 25 | 34 |
| 2 | 7 | 14 | 3 | 3 | 10 | 20 | 15 | 34 | 0 | 0 | 26 | 42 | 15 | 34 | 0 | 0 | 30 | 52 |
| 3 | 3 | 4 | 3 | 3 | 1 | 1 | 4 | 8 | 3 | 3 | 9 | 9 | 9 | 23 | 3 | 3 | 8 | 24 |
| 4 | 4 | 11 | 0 | 0 | 7 | 17 | 5 | 12 | 0 | 0 | 17 | 29 | 6 | 14 | 0 | 0 | 15 | 31 |
| 5 | 0 | 0 | 0 | 0 | 3 | 3 | 1 | 2 | 0 | 0 | 8 | 8 | 6 | 17 | 0 | 0 | 15 | 39 |
| 7 | 7 | 12 | 3 | 4 | 11 | 24 | 24 | 50 | 13 | 27 | 24 | 32 | 24 | 56 | 13 | 36 | 24 | 33 |

**Table 11**
Comparing timetables constructed by the proposed approach with those constructed using EA [12], CG [32], and CP [41] for the gptt instances.

| gptt | Proposed approach | | | | | | EA | | | | | | CG | | | | | | CP | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $z_1$ | $z_2$ | $z_3$ | $z_4$ | $z_5$ | $z_6$ | $z_1$ | $z_2$ | $z_3$ | $z_4$ | $z_5$ | $z_6$ | $z_1$ | $z_2$ | $z_3$ | $z_4$ | $z_5$ | $z_6$ | $z_1$ | $z_2$ | $z_3$ | $z_4$ | $z_5$ | $z_6$ |
| 1 | 3 | 5 | 5 | 15 | 0 | 0 | 3 | 6 | 5 | 15 | 0 | 0 | 5 | 10 | 5 | 15 | 0 | 0 | 5 | 5 | 5 | 15 | 0 | 0 |
| 2 | 5 | 11 | 6 | 17 | 0 | 0 | 7 | 14 | 6 | 21 | 0 | 0 | 6 | 22 | 6 | 22 | 0 | 0 | 5 | 6 | 6 | 22 | 0 | 0 |
| 3 | 2 | 6 | 7 | 17 | 0 | 0 | 2 | 4 | 7 | 22 | 0 | 0 | 6 | 26 | 6 | 16 | 0 | 0 | 6 | 6 | 6 | 16 | 0 | 0 |
| 4 | 5 | 11 | 9 | 21 | 0 | 0 | 6 | 13 | 9 | 29 | 0 | 0 | 6 | 29 | 8 | 29 | 0 | 0 | | | | | | |
| Average improvement | | | | | | | .75 | 1 | 0 | 4.25 | 0 | 0 | 2 | 13.5 | -.5 | 3 | 0 | 0 | 2 | -1.67 | -.33 | 1.33 | 0 | 0 |

**Table 12**
Comparing the average run times of the proposed approach, EA [12], CP [41], and CG [32].

| gptt | Proposed approach (second) | | | EA (minute) | CP (minute) | CG (minute) |
|---|---|---|---|---|---|---|
| | Phase 1 | Phase 2 | Total | | | |
| 1 | 0.20 | 20.45 | 20.66 | 22 | 15 | <60 |
| 2 | 0.38 | 30.58 | 30.95 | 24 | 20 | <60 |
| 3 | 0.23 | 32.16 | 32.39 | 24 | 60 | <60 |
| 4 | 0.66 | 79.39 | 80.05 | 28 | | <60 |

Except for $z_3$ and $z_4$ of instance gbtt2, the proposed approach provides better timetables than either those obtained by EA or those used by the Greek schools. Specifically, it improves all values of $z_1$, $z_2$, $z_5$ and $z_6$ for the timetables of EA and of the Greek high schools for all gbtt instances. It matches the values of $z_3$ and $z_4$ for gbtt1 and gbtt3-5, while it improves them for gbtt7. Thus, the proposed approach provides better quality results. The comparison of its runtime to that of EA further demonstrates the efficiency of the proposed approach. Table 10 shows that, for the gbtt instances, the average run time of the proposed approach (which is 109 second) is negligible in comparison to that of EA (which averages 29 minute). In addition, it indicates that the average time it takes the proposed approach to generate a feasible timetable is very small in comparison to the average time required to improve this solution: 1.74 versus 107.48 second. Moreover, it can be statistically proven that the average runtime of the proposed approach is linearly correlated to the number of classes $n_c$; an expected behavior since the mutation operator is applied to the $n_c$ classes at every iteration. Finally, the average runtime of the proposed approach is correlated to the total number of teachers $\sum_{s\in S} nt_s$.

Table 11 compares the timetables constructed by the proposed approach to those constructed by EA, by the constraint programming (CP) of [41], and by the column generation (CG) of [32], for the gptt instances. CP is a constraint programming approach that is enhanced by tight dynamic bounds and local search techniques that reduce the solution search space. The bounds, computed at various stages of the search, are the solutions to relaxed models, solved using minimum cost matching algorithms. These bounds are used to prioritize the search options of the CP environment whereas the minimum cost matching algorithm serves as an efficient mean to introduce problem domain information to the CP process. CP was run on an HP 9000, 100 MHz and 715 MB of RAM workstation, with a maximum runtime of 1 h per instance.

CG, an iterative procedure, uses a linear programming (LP) relaxation of the original problem. It considers all the variables implicitly. In fact, the legality rules and quality aspects of the weekly schedules of the teachers are present in the model, while their specification remains external to the mathematical model. In each step, CG solves the LP relaxation of the problem using a small number of variables, and determines based on the dual solution, new promising variables that may improve the incumbent. It also uses a branching scheme, based on the fixing of the daily schedule of a teacher, to locate a reasonable integer solution.

None of EA, CG or CP dominates the other two approaches. Thus, the proposed approach is compared to the three of them independently. The proposed approach improves or matches all $z_i,\ i = 1,\ldots,6$, values obtained by EA except for $z_2$ of instance gptt3 where it reaches 6 whereas EA obtains 4. Similarly, the proposed approach improves or matches all $z_i,\ i = 1,\ldots,6$, values obtained by CG and CP except for $z_3$ and $z_4$ of instance gptt3 (where both CG and CP reach 6 and 16 whereas the proposed approach obtains 7 and 17), and $z_3$ for gptt4 where CG reaches 8 while the proposed approach gets 9. These exceptions, do not however reduce the merit of the proposed approach especially that the average improvements it induces in comparison to EA, CG and CP for each of $z_i,\ i = 1,\ldots,6$, values are sizeable in many instances as demonstrated by the last row of Table 11. These average improvements are further amplified when the average runtime of the proposed approach is compared to the runtime of EA, CG and CP, displayed in Table 12. Over the gptt instances, the average runtime of the proposed approach is 41 second versus 24.50 minute for EA, 31.67 minute for CP and less than one hour for CG. The analysis of Table 12 shows, as for the gbtt instances, that the average time it takes the proposed approach to generate a feasible timetable is very small in comparison to the average time required to improve this solution: 0.37 versus 40.64 second. Finally, the average runtime of the proposed approach for gbtt is larger than its counterpart for gptt (i.e., 109.21 versus 41.01 second) since the first set has instances with more classes and teachers than the second set.

## 5. Conclusion

This paper proposes a simulated annealing based approach with a new extended neighborhood structure obtained by performing a series of swaps of pairs of assignments during two time periods. The application of this approach to two benchmark sets of high school timetabling problems shows that this algorithm can compete with other effective approaches; i.e., the new neighborhood structure increases the efficiency and performance of simulated annealing. Other types of neighborhood structures are to be tested with simulated annealing as well as with other meta-heuristics, and their capacity to enhance the performance of the resulting algorithms in solving large-scale timetabling problems is to be determined. Finally, the application of these neighborhood structures in a parallel computing environment is worth investigating.

## Acknowledgments

The authors thank the anonymous referees for their invaluable comments and thank C.N. Moschopoulos, G.N. Beligiannis, and K. Papoutsis for providing them with the benchmark sets. This work was supported by the National Nature Science Foundation of China (Grant No. 60773126), the Province Nature Science Foundation of Fujian (Grant No. A0710023), the academician start-up fund (Grant No. X01109), and the 985 information technology fund (Grant No.

0000-X07204) in Xiamen University. The first author is grateful for their support. This research was partially supported under Kuwait University Research Grant US01/06. The third author is grateful for their support.

## References

[1] S. Abdullah, S. Ahmadi, E.K. Burke, M. Dror, B. McCollum, A tabu-based large neighbourhood search methodology for the capacitated examination timetabling problem, Journal of the Operational Research Society 58 (11) (2007) 1494–1502.
[2] D. Abramson, Constructing school timetables using simulated annealing: Sequential and parallel algorithms, Management Science 37 (1) (1991) 98–113.
[3] D. Abramson, H. Dang, School timetables: A case study in simulated annealing, in: V. Vidal (Ed.), Applied simulated annealing, Lecture Notes in Economics and Mathematics Systems, Springer, Berlin, 1993, pp. 103–124 (Chapter 5).
[4] D. Abramson, M. Krishnamoorthy, H. Dang, Simulated annealing cooling schedules for the school timetabling problem, Asia-Pacific Journal of Operational Research 16 (1999) 1–22.
[5] D. Abramson, M. Randall, A simulated annealing code for general integer linear programs, Annals of Operations Research 86 (1999) 3–21.
[6] S.M. Al-Yakoob, H.D. Sherali, Mathematical programming models and algorithms for a class-faculty assignment problem, European Journal of Operational Research 173 (2) (2005) 488–507.
[7] S.M. Al-Yakoob, H.D. Sherali, A mixed-integer programming approach to a class timetabling problem: A case study with gender policies and traffic considerations, European Journal of Operational Research 180 (3) (2007) 1028–1044.
[8] R. Alvarez-Valdes, E. Crespo, J.M. Tamarit, Design and implementation of a course scheduling system using tabu search, European Journal of Operational Research 137 (2002) 512–523.
[9] A.S. Asratian, D. de Werra, A generalized class-teacher model for some timetabling problems, European Journal of Operational Research 143 (2002) 531–542.
[10] P. Avella, B. D'Auria, S. Salerno, I. Vasil'ev, A computational study of local search algorithms for Italian high-school timetabling, Journal of Heuristics 13 (2007) 543–556.
[11] Z.N. Azimi, Hybrid heuristics for examination timetabling problems, Applied Mathematics and Computation 163 (2) (2005) 705–733.
[12] G.N. Beligiannis, C.N. Moschopoulos, G.P. Kaperonis, S.D. Likothanassia, Applying evolutionary computation to the school timetabling problem: The Greek case, Computers and Operations Research 35 (4) (2008) 1265–1280.
[13] T. Birbas, S. Daskalaki, E. Housos, Timetabling for Greek high schools, Journal of the Operational Research Society 48 (1997) 1191–1200.
[14] B. Bullnheimer, An Examination scheduling model to maximize students' study time, in: E. Burke, M. Carter (Eds.), Practice and Theory of Automated Timetabling II (LNCS 1408), Springer-Verlag, Berlin, 1998, pp. 8–91.
[15] E.K. Burke, S. Petrovic, Recent research directions in automated timetabling, European Journal of Operational Research 140 (2002) 266–280.
[16] E.K. Burke, J.P. Newall, Solving examination timetabling problems through adaption of heuristic orderings, Annals of Operations Research 129 (2004) 107–134.
[17] E.K. Burke, B. McCollum, A. Meisels, S. Petrovic, R. Qu, A graph-based hyper-heuristic for educational timetabling problems, European Journal of Operational Research 176 (2007) 177–192.
[18] E.K. Burke, A. Eckersley, B. McCollum, S. Petrovic, R. Qu, Hybrid variable neighbourhood approaches to university exam timetabling, European Journal of Operational Research, submitted for publication.
[19] M.P. Carrasco, M.V. Pato, A Potts neural network heuristic for the class/teacher timetabling problem, Applied Optimization Metaheuristics: Computer Decision-making Book, Kluwer Academic Publishers, 2004. pp. 173–186.
[20] M.P. Carrasco, M.V. Pato, A comparison of discrete and continuous neural network approaches to solve the class/teacher timetabling problem, European Journal of Operational Research 153 (2004) 65–79.
[21] P. De Causmaecker, P. Demeester, G.V. Berghe, A decomposed metaheuristic approach for a real-world university timetabling problem, European Journal of Operational Research (2008), doi:10.1016/j.ejor.2008.01.043.
[22] B. McCollum, A perspective on bridging the gap between research and practice in university timetabling, in: E. Burke, H. Rudova (Eds.), Practice and Theory of Automated Timetabling VI, (LNCS 3867), Springer-Verlag, Berlin, 2007, pp. 3–23.
[23] B. McCollum, P. McMullan, B. Paechter, R. Lewis, A. Schaerf, L. Di Gaspero, A.J. Parkes, R. Qu, E. Burke, Setting the research agenda in automated timetabling: The second international timetabling competition, INFORMS Journal of Computing (2009), doi:10.1287/ijoc.1090.0320.
[24] S. Daskalaki, T. Birbas, E. Housos, An integer programming formulation for a case study in university timetabling, European Journal of Operational Research 153 (2004) 117–135.
[25] S. Daskalaki, T. Birbas, Efficient solutions for a university timetabling problem through integer programming, European Journal of Operational Research 160 (2005) 106–120.
[26] E. Eley, Ant algorithms for the exam timetabling problem, in: Burke, E.K., Rudova, H. (Eds.), Practice and Theory of Automated Timetabling VI: Selected Papers from the 6th International Conference, (LNCS 3867), 2007, pp. 364–382.
[27] R. Gonzalez-del-Campo, F. Saenz-Perez, Programmed search in a timetabling problem over finite domains, Electronic Notes in Theoretical Computer Science 177 (2007) 253–267.
[28] D. Johnson, Timetabling university examinations, Journal of the Operational Research Society 41 (1990) 39–47.
[29] H.Y. Lee, Y.C. Lin, A decision support model for scheduling exhibition projects in art museums, Expert Systems with Applications (2009), doi:10.1016/j.eswa.2009.03.003.
[30] R. Lewis, A survey of metaheuristic-based techniques for university timetabling problems, OR Spectrum 30 (2007) 167–190.
[31] S.K. Mirrazavi, S.J. Mardle, M. Tamiz, A two-phase multiple objective approach to university timetabling utilizing optimisation and evolutionary solution methodologies, Journal of the Operational Research Society 54 (2003) 1155–1166.
[32] K. Papoutsis, C. Valouxis, E. Housos, A column generation approach for the timetabling problem of Greek high schools, Journal of the Operational Research Society 54 (2003) 230–238.
[33] S. Petrovic, E.K. Burke, University timetabling, Handbook of Scheduling: Algorithms, Models, and Performance Analysis, CRC Press, Boca Raton, FL, 2004 (Chapter 45).
[34] R. Qu, E.K. Burke, B. McCollum, L.T. Merlok, S.Y. Lee, A survey of search methodologies and automated approaches for examination timetabling, Journal of Scheduling 12 (1) (2009) 55–89.
[35] M. Randall, D. Abramson, C. Wild, A general meta-heuristic based solver for combinatorial optimisation problems, Computational Optimization and Applications 20 (2) (2001) 185–210.
[36] K. Socha, M. Sampels, M. Manfrin, Ant algorithms for the university course timetabling problem with regard to the state-of-the-art, in: Proceedings of EvoCOP 2003, 3rd European Workshop on Evolutionary Computation in Combinatorial Optimization (LNCS 2611), 2003, pp. 334–345.
[37] K.A. Smith, D. Abramson, D. Duke, Hopfield neural networks for timetabling: formulations, methods, and comparative results, Computers and Industrial Engineering 44 (2) (2003) 283–305.
[38] J. Thompson, K.A. Dowsland, A robust simulated annealing based examination timetabling system, Computers and Operations Research 25 (1998) 637–648.
[39] J. Thompson, K.A. Dowsland, General cooling schedules for a simulated annealing based timetabling system, in: E. Burke, P. Ross (Eds.), Practice and Theory of Automated Timetabling I, (LNCS 1153), Springer-Verlag, Berlin, 1996, pp. 345–363.
[40] H. Ueda, D. Ouchi, K. Takahashi, T. Miyahara, Comparisons of genetic algorithms for timetabling problems, Systems and Computers in Japan 35 (7) (2004) 691–701.
[41] C. Valouxis, E. Housos, Constraint programming approach for school timetabling, Computers and Operations Research 30 (2003) 1555–1572.
[42] D. de Werra, A.S. Asratian, S. Durand, Complexity of some special types of timetabling problems, Journal of Scheduling 5 (2002) 171–183.
[43] J. Wood, D. Whitaker, Student centred school timetabling, Journal of the Operational Research Society 49 (1998) 1146–1152.