Available online at www.sciencedirect.com
SCIENCE DIRECT®

Computers & Operations Research 32 (2005) 1761 – 1776

computers & operations research

www.elsevier.com/locate/dsw

A two-phase heuristic evolutionary algorithm for personalizing course timetables: a case study in a Spanish university

Ricardo Santiago-Mozos, Sancho Salcedo-Sanz*, Mario DePrado-Cumplido, Carlos Bousoño-Calzón

Department of Signal Theory and Communications, Universidad Carlos III de Madrid, Leganes, Madrid 28911, Spain

Abstract

This paper presents, as a case study, the application of a two-phase heuristic evolutionary algorithm to obtain personalized timetables in a Spanish university. The algorithm consists of a two-phase heuristic, which, starting from an initial ordering of the students, allocates students into groups, taking into account the student's preferences as a primal factor for the assignment. An evolutionary algorithm is then used in order to select the ordering of students which provides the best assignment.

The algorithm has been tested in a real problem, the timetable of the Telecommunication Engineering School at *Universidade de Vigo* (Spain), and has shown good performance in terms of the number of constraints fulfilled and groups assigned to students.

© 2004 Elsevier Ltd. All rights reserved.

Keywords: Timetabling; Combinatorial optimization; Heuristics; Evolutionary algorithms

1. Introduction

Timetabling is a problem that most universities in the world must face year after year [1–4]. The basic timetabling problem in a university consists of finding time slots for a set of events (exams or subjects for example) so that students can attend all respective events [4–6]. The definition of a timetabling problem usually differs from one institution to another, as every university has different necessities and peculiarities for subject registration, exams or classes, which in many cases vary from year to year [7,8]. Thus, a large number of variants of the timetabling problem can be found in the literature, which differ from each other both in the type of institution involved

* Corresponding author. Tel.: +34-91-624-5973; fax: +34-91-624-8749.
E-mail address: sancho@tsc.uc3m.es (S. Salcedo-Sanz).

0305-0548/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2003.11.030

(universities, high schools) and in the type of constraints considered [7]. Two types of constraints can be defined in every timetabling problem: first, the constraints which are basic for the feasibility of the timetable obtained are normally called *hard constraints*. Second, the constraints which do not affect the feasibility of the solution found, but their fulfilment makes it more appropriate in terms of some defined criteria. These constraints are usually called *soft constraints* [6].

There is a vast amount of literature devoted to the timetabling problem, where several algorithms and heuristics have been proposed, each tackling a different aspect of the problem. For instance, there are a number of interesting surveys of existing timetabling methods and applications [4,5,9–11]. These works represent the effort of the scientific community on timetabling over the last 20 years. Among the methods used to solve the timetabling problem we highlight the *Constraint-based* approaches, which have been used to solve some instances of the timetabling problem in [12–14] or [15]; the *Sequential* methods, mainly graph coloring algorithms, [4,7] which have been used to tackle timetabling problems in the last few years; and above all the so called *emergent algorithms* such as heuristics and meta-heuristics (genetic algorithm, tabu search, simulated annealing), which have been applied to timetabling for searching large classroom scheduling [16], for managing high-school timetables [17] or in some other works like [6,18–21,32,33]. In the last few years there has been increasing research work on the application of these emergent algorithms to timetabling problems.

Finally, we would like to highlight a few works which consider some preferences of students as soft constrains of the problem. For example Rudová et al. [22,23] propose an approach which uses *annotations* in variables in order to solve the problem's hard constraints and manage the student's preferences at the same time. In [24], an approach to soft constraints management in timetabling problems using *constraint logic programming* is discussed. We also take into consideration the work by Paechter et al. [25], in which they debate certain preferences of students when solving the timetabling of an entire university by means of an evolutionary algorithm.

In this paper we present the application of a two-phase search heuristic evolutionary algorithm in a real university timetabling problem: The two-phase heuristic has been specifically designed for timetabling, and is able to obtain feasible personalized timetables starting from a given ordering of students. The performance of the two-phase heuristic depends on the initial ordering of the students, so the approach is completed by an evolutionary algorithm (EA). This looks for the student ordering and provides a better solution in terms of student's preferences and other characteristics of the timetables (mainly *compactness* and *non-priority subject assignment*, which will be defined below).

This algorithm was applied to a real timetabling problem in a Spanish university: School of Telecommunications Engineering, *Universidade de Vigo* (Galicia, Spain), where the personalized timetables for 1301 students of the school were obtained by means of our algorithm. We will show that it was able to find complete personalized and feasible timetables, where the majority of student's petitions were granted.

The structure of the article is as follows: Section 2 briefly introduces the problem and its significance for a large range of universities. We also provide a model of the organization of teaching in Spanish faculties which is used to define the problem. In this section the complete mathematical definition of the problem is also given, describing the main constraints and goals of the problem. Section 3 describes and analyzes the proposed algorithm, whereas Section 4 shows the results of the simulations performed in order to test it in a real case. Section 5 concludes the paper and offers some final remarks.

2. Problem definition

2.1. Personalizing timetables

In this section we briefly describe the importance of personalizing timetables based on the case of some Spanish universities, and we also formulate the problem. First, we describe how the teaching is organized in several Spanish universities. Second, we elaborate a general description of the problem and go on to give a mathematical description of it.

The majority of Spanish faculties offer the possibility of freely choosing the number of subjects a student wishes to study over a year. It is common for students to choose subjects belonging to different courses (e.g. subjects belonging to the first and second course of a given degree), usually in science and technology faculties.[1]

In order to focus the problem, we consider the following scenario, which models the organization of teaching in a large number of faculties and schools in Spain:[2]

(1) The faculty assigns several groups to every subject, every group has a fixed timetable for classes. Every group has been previously assigned to rooms or laboratories, and there is a maximum number of students per classroom or laboratory.[3]

(2) The groups are already scheduled to lecturers, i.e. staffing constraints are already satisfied.[4]

(3) Every student is allowed to register in every subject the faculty offers (in different courses if desired), and they are also allowed to choose a preferential group for each subject. Note that for every subject, the classes have a different timetable depending on the group.

(4) Every student must choose a set of "priority" subjects belonging to *the same course*, theoretical or practical (laboratory) subjects, which will be assigned with priority over subjects belonging to different courses.

(5) Since every group is restricted to a maximum number of students, the assignment of the desired group to a given student is not always possible, and therefore another group should be assigned.

A large number of faculties can be modelled by means of the scenario described above. Considering this model, the objective of every faculty is to assign a feasible and personalized weekly timetable for the whole course[5] to every student. By *personalized* timetable we mean: first, that the timetable should include all the subjects that a given student has chosen as priority subjects, including as many non-priority subjects as possible, and second, the students should be assigned to the group they have chosen if possible, or in another feasible group if the first election cannot be provided. The students must be able to attend all classes, i.e. the students cannot be assigned to

[1] We focus our attention on technology schools, due to the application we present in this paper consists in obtaining the personalized timetables for students of telecommunication engineers school; however, the approach is extensible to any other type of faculty or school with similar characteristics to those we consider in this paper.

[2] Note that this scenario demarcates our problem from traditional timetabling problems.

[3] This means that groups has also a maximum number of students allowed. This capacity is different depending on whether it is a theory group or a laboratory group.

[4] In the majority of Spanish faculties and schools this issue is previously solved by the direction of the school together with lecturers.

[5] Note that the timetable assigned consists of the classes a given student has from Monday to Friday, from 8 o'clock in the morning until 8 o'clock in the evening, during the entire course.

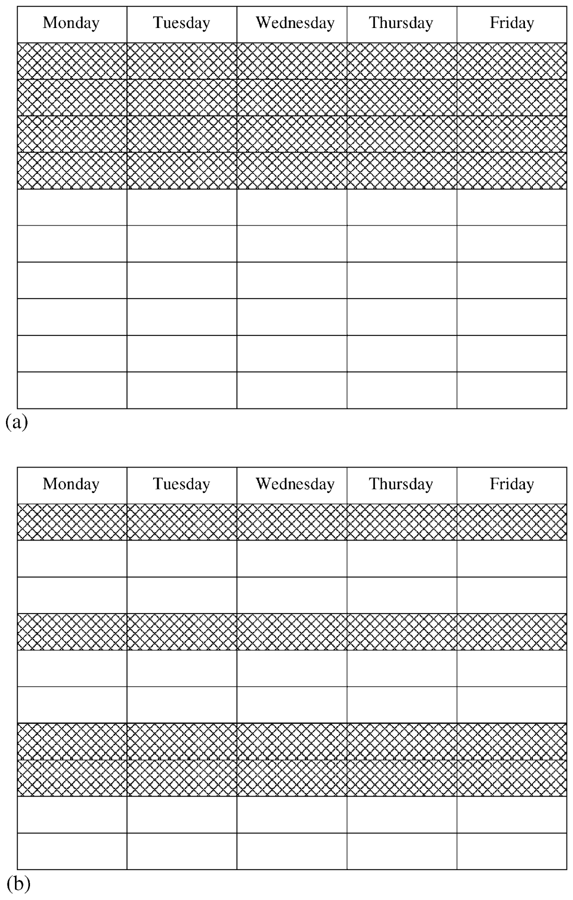

Fig. 1. (a) Example of a compact timetable. Shadowed zones stands for assigned groups. $C_i = 0$ in this example. (b) Example of a non-compact timetable. $C_i = 20$ in this example.

groups with overlapped timetables. This is a hard constraint that must always be fulfilled. Finally, we also consider as a goal of the problem that every student's timetable must be as *compact* as possible. The concept of compactness (C_i) is defined as the total number of free hours between two assigned groups in a timetable. Fig. 1(a) and (b) shows an example of a compact timetable and a non-compact one, respectively. In Appendix A, it is shown how to calculate the compactness of a given timetable using Fig. 1(a) and (b).

Note that the timetabling problem we face is defined as being focused on obtaining the best commodities for the students, and its objectives and constraints are different from the traditional timetabling problem.

2.2. Mathematical formulation of the problem

Table 1 defines the variables used hereafter in the mathematical formulation of the problem and in the algorithm description for solving it. The problem formulation is presented below, in Eqs. (1)–(6).

Table 1
Notation

Variable	Definition
Input	
N_s	Total number of students
N_b	Total number of subjects
N_{g_j}	Total number of groups in subject j, $j \in \{1,\dots,N_b\}$
G_{jk}	Maximum number of students in group k, $k \in \{1,\dots,N_{g_j}\}$ of subject j, $j \in \{1,\dots,N_b\}$
R_{ij}	Registration matrix. Binary matrix in which a 1 in position (i,j) means that student i, $i \in \{1,\dots,N_s\}$ has registered in subject j, $j \in \{1,\dots,N_b\}$
P_{ij}	Priority matrix. Binary matrix in which a 1 in position (i,j) means that student i, $i \in \{1,\dots,N_s\}$ has chosen subject j, $j \in \{1,\dots,N_b\}$ as priority subject. No subjects belonging to different courses can be chosen as priority subjects
M_{ij}	Matrix of non-priority subjects. It is a binary matrix defined as $M_{ij} = R_{ij} - P_{ij}$
$F_{jkj'k'}$	Binary matrix of feasibility among groups. Given two groups k and k', $k \in \{1\dots N_{g_j}\}$, $k' \in \{1,\dots,N_{g_{j'}}\}$ belonging to subjects j, $j \in \{1,\dots,N_b\}$ and j', $j' \in \{1,\dots,N_b\}$, $F_{jkj'k'}$ is 0 if there is not incompatibility between groups k and k', and 1 if there is incompatibility between them
D_{ijk}	Binary matrix of preferences in which a 1 in position (i,j) means that student $i \in \{1,\dots,N_b\}$ has chosen group k, $k \in \{1,\dots,N_{g_j}\}$ in subject j, $j \in \{1\dots N_b\}$
Output	
V_{ijk}	Assignment matrix: for each student i, $i \in \{1,\dots,N_s\}$, subject j, $j \in \{1,\dots,N_b\}$ and group k, $k \in \{1,\dots,N_{g_j}\}$, V_{ijk} is 1 if the group has been assigned, and 0 if not
C_i	Vector of timetable compactness. For every student i, $i \in \{1,\dots,N_s\}$ C_i is a measure of the student's timetable compactness. This vector is calculated based on the assignment matrix V_{ijk} (see Appendix A for an example)

The first objective (objective (1)) is to minimize the total number of non-assigned subjects. We also desire that the obtained timetables be as compact as possible. This is achieved by considering Eq. (2) as an objective of the problem (objective (2)). Finally, timetables obtained should satisfy the maximum of students preferences (objective (3)). Problem constraint (4) stands for assigning all the priority subjects of students, problem constraint (5) ensures the compatibility among groups and finally, problem constraint (6) restricts the number of students per group.

(a) Non-priority assignment requirements:

$$\min\left(\sum_{i=1}^{N_s}\sum_{j=1}^{N_b}\left(M_{ij}\left(1-\sum_{k=1}^{N_{g_j}}V_{ijk}\right)\right)\right). \tag{1}$$

(b) Compactness requirements:

$$\min\left(\sum_{i=1}^{N_s}C_i\right). \tag{2}$$

(c) Group preference requirements:

$$\max\left(\sum_{i=1}^{N_s}\sum_{j=1}^{N_b}\sum_{k=1}^{N_{g_j}} V_{ijk}D_{ijk}\right). \tag{3}$$

(d) Feasibility constraint I (priority assignment requirement):

$$\sum_{i=1}^{N_s}\sum_{j=1}^{N_b}\sum_{k=1}^{N_{g_j}} V_{ijk}R_{ij}P_{ij} = \sum_{i=1}^{N_s}\sum_{j=1}^{N_b} R_{ij}P_{ij}. \tag{4}$$

(e) Feasibility constraint II (compatibility among groups):

$$\sum_{i=1}^{N_s}\sum_{j=1}^{N_b}\sum_{k=1}^{N_{g_j}}\sum_{j'>j}^{N_b}\sum_{k'>k}^{N_{g_{j'}}} V_{ijk}V_{ij'k'}F_{jkj'k'} = 0. \tag{5}$$

(f) Feasibility constraint III (maximum number of students per group):

$$\sum_{i=1}^{N_s} V_{ijk} \leqslant G_{jk} \quad \forall j,k. \tag{6}$$

3. Description of the algorithm

This section describes in detail the algorithm developed in this paper. We propose a two-phase heuristic evolutionary approach, where the two-phase heuristic obtains feasible personalized timetables; and the EA improves the solutions in terms of compactness of timetables and low-priority subjects assignment. The two-phase heuristic algorithm is described in Section 3.1, whereas the characteristics of the EA used are described in Section 3.2. Note that the structure of our algorithm (a priority list evolved with an evolutionary algorithm together with a scheduling engine (the two-phase heuristic in our case)) is well known in scheduling problems, see for example the work by Fang et al. [26] for the open-shop scheduling problem, a problem quite close to timetabling, and the work by Lai et al. [27] for the frequency assignment problem.

3.1. The two-phase heuristic search

The two-phase heuristic used in this approach involves two searching procedures (two-phases) each one guaranteeing the fulfilment of a different set of constraints.

3.1.1. Priority subjects assignment

The first task must be to assign groups in a feasible way for the subjects that students have chosen as being priorities. In order to do this, we start from an ordering $\pi(i)$, $i = 1,\ldots,N_s$ of students. The assigning matrix V_{ijk} is set to 0 for all students (no assignment); then for every student $\pi(i)$, the first attempt is assigning priority subjects to the groups he/she has chosen, i.e. $V_{\pi(i)jk}=1$ if $P_{\pi(i)j}=1$ and $D_{\pi(i)jk}=1$, where j is the subject the student $\pi(i)$ is going to follow and k is the group he has chosen for that particular subject. The assignment will be feasible only if $F_{jkj'k'}=0$ whether $V_{\pi(i)jk}=1$ and $V_{\pi(i)j'k'}=1$ (problem constraint (e)), and also none of the selected groups is full over, i.e. $\sum_{i=1}^{N_s} V_{ijk} \leqslant G_{jk}$ (problem constraint (f)). If $F_{jkj'k'}=1$ or $\sum_{i=1}^{N_s} V_{ijk} > G_{jk}$, *all* the groups assigned to the student $\pi(i)$ are removed and reassigned into uncompleted groups in such a way that $F_{jkj'k'}=0$. This reassignment follows a sequential algorithm, checking first the groups with more similar timetables to the one chosen. However, note that in this last case the student will be forced to study in groups they did not choose, thus, these assignments do not contribute to the increase in the preferences requirement term (Eq. (3)). Note also that the three hard constraint of feasibility between priority subjects (constraints (d)–(f)) are fulfilled, due to $F_{jkj'k'}=0$ if $V_{ijk}=1$ and $V_{ij'k'}=1$ and $\sum_{i=1}^{N_s} V_{ijk} \leqslant G_{jk}$. Finally, recall that only subjects belonging to *the same* course can be selected as priority subjects, so it will always be possible to achieve an assignment V_{ijk} which fulfils problem constraint (d). Summing up, the first heuristic proposed can be described in pseudo-code as:

Pseudo-code of the first heuristic

```
for every student $\pi(i)$:
    for every subject j:
        if($P_{\pi(i)j}=1$ and $D_{\pi(i)jk}=1$)
            **Assign_priority_groups**($\pi(i), j, k$);
        end(if)
    end(subject j)
        if(group full over or infeasible assignment)
            **Reassign_priority_groups**($\pi(i)$);
        end(if)
end(student $\pi(i)$)
```

3.1.2. Non-priority subjects assignment

Once the priority assignment has been performed, a second heuristic manages the non-priority subjects assignment, in the following way: the non-priority subjects fulfil the condition $M_{ij}=1$. For these subjects, we again start from the ordering $\pi(i)$ of students. We firstly attempt to assign the subject to the desired group k such as $D_{ijk}=1$. We check that the group is not full and that the assignments are feasible for a given student. The conditions to be fulfilled are again $F_{jkj'k'}=0$ whether $V_{\pi(i)jk}=1$ and $V_{\pi(i)j'k'}=1$ (problem constraint (e)), and $\sum_{i=1}^{N_s} V_{ijk} \leqslant G_{jk}$ (problem constraint (f)). If there is not a feasible group or if all the groups are already full, then we try to assign the subject to another feasible group, again starting from groups with timetable more similar to the one desired. In the case that no feasible option can be found, or all feasible groups are full, the subject is not assigned. The second heuristics in pseudo-code is the following:

Pseudo-code of the second heuristic

```
for every student $\pi(i)$:
  for every subject j:
    if($M_{\pi(i)j} = 1$ and $D_{\pi(i)jk} = 1$)
      **Assign_non-priority_groups**($\pi(i), j, k$);
    end(if)
    if(group full over or infeasible assignment)
      **Reassign_subject**($\pi(i), j$);
    end(if)
  end(subject j)
end(student $\pi(i)$)
```

3.1.3. Analysis of the two-phase heuristic

The heuristic presented in this paper for timetabling has two different parts (two-phases); the heuristic for the priority subjects and the heuristic for the non-priority subjects. Focusing on the first heuristic, and due to its structure, the first students to be managed will achieve the desired groups for priority subjects, whereas the last students will probably have to study some subjects in non-desired groups. In addition, once the first heuristic has assigned all the priority subjects, the number of vacant places in groups will have decreased, and the number of unassigned subjects will depend completely on the initial ordering $\pi(i)$ of students.

Another important point to be noted is that the two-phase heuristic search will produce good timetables, i.e. timetables in which the problem's hard constraints are fulfilled. However, these solutions can be improved by means of an evolutionary algorithm, which produces very high-quality solutions in terms of soft constraints fulfilment.

3.2. The evolutionary algorithm

The concept of *evolutionary algorithm* (EA) is based on natural evolution. In nature, the individuals constituting a population adapt to the environment in which they live. The fittest individuals have the highest probability of survival and tend to increase in number, while the less fit individuals tend to die out. This survival of the fittest principle is the idea behind EAs [28].

EAs maintain a *population* of individuals, each of which represents a specific solution to the given optimization problem. Starting from a random generated population, a process of evolution is simulated. The main components of this process are the operators of *selection*, *crossover* and *mutation*, which emulate the random changes occurring in nature. They will be explained in detail in this section. After a number of generations, highly fit individuals will emerge corresponding to good solutions to the given optimization problem.

In this paper we use an EA for improving the solutions found by the two-phase heuristic procedure. Every individual of the EA is a string of numbers which encodes a permutation $\pi(i)$, $i = 1, \ldots, N_s$, representing the ordering of students which will be used by the two-phase heuristic algorithm. The population of the EA is formed by ξ individuals (strings) in which the genetic operators, selection, crossover and mutation, are applied. A fitness value is associated to every individual in the population. We define this fitness as a measure of the problem's requirements. The population is then evolved by means of the application of genetic operators to it. The pseudo-code of the proposed EA is the following:

Pseudo-code of EA

```
Init_Population($\pi(i)$)
fitness_calculation
 While ($Ngenerations \leqslant 100$)
   Selection
   Crossover (PMX)
   Mutation
   fitness_calculation
   get_better_individual
   $Ngenerations{+}{+}$;
 end(while)
```

where N stands for the number of generations (we stop the algorithm after 100 generations), and the best individual of the current population is always passed to the next generation (elitism operator).

3.2.1. Selection operator

The selection operator is responsible for choosing which individuals will survive for the next generation of the EA. Among the different types of selection procedures existing [29], we have chosen the one known as *roulette wheel*, in which the probability of an individual to be selected for the next generation ($P(i)$) depends on its current fitness value:

$$P(i) = \frac{f_i}{f_T}, \tag{7}$$

where f_i is the fitness value associated with individual i and f_T is the total fitness of the population, which is defined as $f_T = \sum_{i=1}^{\xi} f_i$.

Thus, it is probable that the fittest individuals receive a larger number of samples in the next generation than individuals with less associated fitness values.

3.2.2. Crossover operator

The crossover operator has been described as the key to the EA's power [28,30], as it promotes structured yet randomized information exchange between individuals. However, if the crossover operator is applied to every individual in the population, there will be a discontinuity from the previous to the present populations, as none of the individuals of the population from the previous generation will be retained in the new one. In order to avoid this, a *crossover probability* α_x is defined. It has been suggested that $\alpha_x < 1$, and the range of values used usually lies between $\alpha_x = 0.5–0.6$ [31]. The pseudo-code of the Crossover operator used in this paper is as follows:

Pseudo-code of Crossover operator

```
Couple all individuals, at random.
  for(each couple)
    if(random_variable(0,1) $\leqslant$ 0.6)
      Perform_Crossover(PMX);
    end(if)
  end(for)
```

The application of traditional crossover to a population of permutations, as in our case, would produce infeasible individuals, that would not represent a permutation after the crossover operation. One technique that has been used extensively to avoid similar problems, is to use a more sophisticated crossover operator, known as *partially matched crossover* (PMX) [27]. In PMX, once the individuals have been coupled at random, two points in a string are randomly chosen, and the portions of individuals are exchanged. In addition, if PMX encounters a conflict, i.e. a duplicate number in an individual, it will resolve this conflict by swapping the corresponding value between the individuals. Fig. 2 shows an example of how the PMX operator works: After the crossover of the selected parts, there is a conflict in parent 1 (duplicate numbers) in the first location (a 3), second location (a 2) and sixth location (a 6). In parent 2 there is a conflict in the second position (a 5), sixth position (a 1) and seventh position (a 4). Then, every position in which there is a conflict, is substituted by the corresponding swapped value. For example, in parent 1, the first location (a 3) is substituted by the value which was swapped with the 3, in this case a 4. In second position, conflict value (a 2) is substituted by a 1 and in sixth position, the 6 is substituted by a 5. In parent 2, in the second position the 5 is substituted by a 6, in sixth position the 1 is substituted by a 2 and in seventh position the 4 is substituted by a 3.

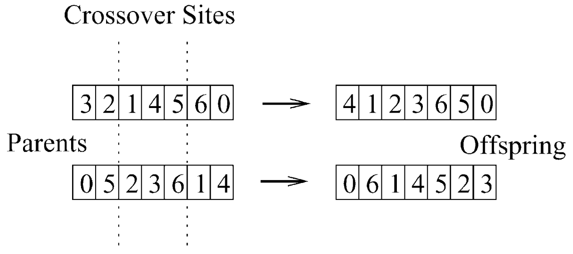

Fig. 2. Partially matched crossover (PMX) example.

Pseudo-code of PMX operator

```
Given two parents for crossover:
Select two swapping points randomly.
Swap the part of the parents between the swapping points.
  for(the two parents)
    for(all positions but the swapped)
      if(any repeated values)
      Substitute value for the corresponding swapped value;
      end(if)
    end(for)
  end(for)
```

3.2.3. Mutation operator

After the crossover operation described above, every single individual in the population may undergo a further random change with a small probability α_m. This change consists of choosing two points in the string of numbers representing an individual and swapping the values in them. Note that this operation does not produce infeasible individuals.

3.2.4. Fitness calculation

The fitness function of our EA includes three terms, each representing a problem's requirement. Note that the first two requirements (a) and (b) involve the minimization of an expression, whereas the EA searches for the maximum of the fitness function. Thus, we define it as

$$F = \left(K - \sum_{i=1}^{N_s} \sum_{j=1}^{N_b} a_j \left(M_{ij} \left(1 - \sum_{k=1}^{N_{g_j}} V_{ijk} \right) \right) - b \sum_{i=1}^{N_s} C_i \right) + c \sum_{i=1}^{N_s} \sum_{j=1}^{N_b} \sum_{k=1}^{N_{g_j}} V_{ijk} D_{ijk}, \tag{8}$$

where K is an upper bound of

$$\left(\sum_{i=1}^{N_s} \sum_{j=1}^{N_b} a_j \left(M_{ij} \left(1 - \sum_{k=1}^{N_{g_j}} V_{ijk} \right) \right) + b \sum_{i=1}^{N_s} C_i \right)$$

a_j and b are penalty terms for not assigning a subject and for lack of compactness of the timetables, respectively, and c is a premium for assigning a given subject to the student's desired group.

4. Experiments and results

In this paper we face a real problem: the assignment of personalized timetables in a school of telecommunications engineers. Specifically, the presented algorithm was used to assign personalized timetables to 1301 students of the School of Telecommunications Engineers, *Universidade de Vigo*

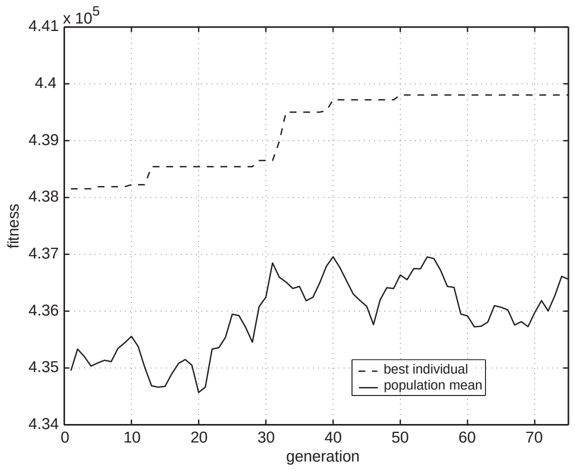

Fig. 3. Evolution of the fitness function (population mean and best individual).

(Galicia, Spain), in October 2002. In this school, students can choose among 108 subjects, including theoretical subjects and practical subjects (laboratories). The teaching organization follows the model described in Section 2, and our algorithm is directly applicable, without any changes. In the registration form, students choose the number of subjects they wish to follow, and the ones that are priority subjects, belonging to the same course.

With the data collected in the registration process, we construct all the matrices and parameters needed for running the algorithm, i.e. matrices R_{ij}, P_{ij}, M_{ij}, D_{ijk}, $F_{jj'kk'}$, G_{jk}, and parameters N_s, N_b and N_{g_j}. We found that the best election of penalty terms for not assigning a non-priority subject (a_j) depend on the subject.[6] The best results were obtained with a_j in a range between 1000 and 1300, the penalty for lack of compactness $b = 15$ and the premium for assigning a subject to a preferred group $c = 20$. These values are used in the calculation of the fitness associated to every individual in the EA (see Eq. (8)). The EA's parameters in all simulations were fixed to $\alpha_x = 0.6$ and $\alpha_m = 0.01$, with a population of 50 individuals. We programmed our algorithm in C++, using PostgreSQL as database. We use a SUN SPARK 2/480 MHz for running the simulations. The approximated computation time in that simulation platform was about 3 h.

The total number of subjects requested by the students in the registration process ($\sum_{i=1}^{N_s} \sum_{j=1}^{N_b} R_{ij}$) was 13,305, including theoretical and practical (laboratory) subjects. We ran the algorithm with the parameters referred above; it was able to assign 13,175, 12,102 of which were allocated in the desired groups, and the rest in other groups. This means that our algorithm assigns over 99% of the total requested subjects, and over 90% of them in the desired groups. Fig. 3 shows the evolution of the fitness (population mean and best individual) against the generations. Note that the algorithm converge to the best solution found in about 50 generations. Fig. 4 depicts the evolution of the lack of compactness, (mean population and best individual) of timetables achieved. Note that due to the penalty for not assigning a subject is much larger than the penalty due to lack of compactness, better

[6] For example, we penalized more failing to assign a laboratory subject than a theoretical subject.

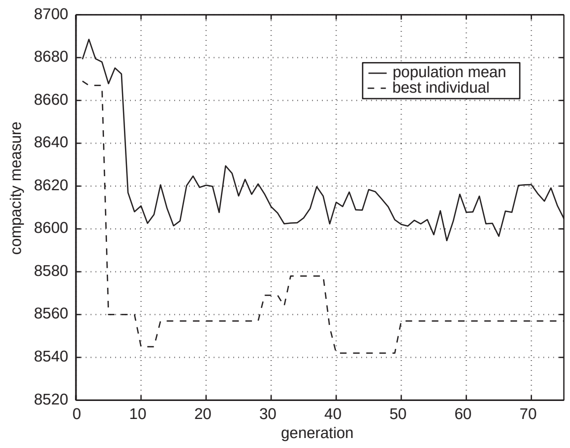

Fig. 4. Evolution of the total compactness of timetables obtained (population mean and best individual).

Table 2
Comparison of the results obtained by different ordering algorithms

Algorithm	Total non-priority assigned subjects	Subjects assigned in the desired groups	Total timetables compactness
EA	13,175	12,102	8558
Greedy	13,151	12,047	8610
Random	13,096	11,915	8670

solutions in terms of number of assignments may have worse properties of compactness. However, as can be seen in Fig. 4, our approach is able to control it.

In order to check what the effect is of using an EA as global search algorithm, we have compared the results obtained using it against the results found with a greedy algorithm for ordering the students, and also against the results obtained with a random ordering. [7]

The greedy algorithm consists of assigning first the students with a larger number of subjects. We have run 100 times the algorithm with a random ordering, keeping the best solution found. Table 2 shows a comparison of the solutions found by the different ordering algorithms. It is easy to see that the best solution is found by the EA, in terms of non-priority subjects assigned and subjects assigned to preferred groups. The compactness of the timetables is also better in the solution found by the EA than that obtained by the other two algorithms.

[7] Recall that the performance of the algorithm depends on the initial ordering ($\pi(i)$) of the students, as was pointed out in Section 3.1.3. Note also that the local search heuristics are the same for the three algorithms, what ensures that the problem's constraints will be fulfilled.

5. Conclusions

In this paper we have presented, as a case study, the application of a two-phase heuristic evolutionary algorithm to obtain personalizing timetables in some courses of a Spanish university. We tackle the problem of assigning a feasible and personalized timetable to every student in a faculty or school. Thus, students are allowed to choose a set of priority subjects which will be always assigned, and also a preferred group for following a given subject. Other objectives such as compactness of timetables constructed and non-priority subjects assignment have also been considered.

The algorithm used is formed by a two-phase heuristic for solving the problem constraints and an evolutionary algorithm for improving the quality of the solutions found. We have applied it to a real problem, consisting of assigning personalized timetables to 1301 students of the School of Telecommunications Engineers, *Universidade de Vigo* (Galicia, Spain), in October 2002. We have obtained very good results in terms of non-priority subjects assignment (over 99%), compactness of timetables and assignments of preferred groups to students. Thus, this paper is a good example of the application of emergent techniques such as evolutionary algorithms and heuristic search to real-life problems.

Acknowledgements

The authors would like to thank Dr. José Luis Rojo-Álvarez, Universidad Carlos III de Madrid and Dr. Georgios Theodoropoulos, University of Birmingham, for their interesting comments and hints. We would like to express our gratitude to Miss Angela Richardson, University of Birmingham, for her useful correction of the paper. We wish also thank people from School of Telecommunications at *Universidade de Vigo* for their assistance and courtesy. The authors really appreciate the effort of unknown referees reviewing the paper, their useful comments and indications, which have contributed to improve the quality of this article.

Appendix A.

A.1. Calculation of the compactness of a timetable

We use the following expression in order to calculate the compactness of the timetable of a given student i:

$$C_i = \sum_{d=1}^{5} \sum_{h=1}^{12} W_{dh}(FR)_{dh}, \tag{9}$$

where index d stands for the day of the week, index h stands for the hour of the day, $W_{dh} = 1$ if h is between the first and last hour when the student has a class (0 otherwise), and $(FR)_{dh} = 1$ is h corresponds to a free hour and 0 otherwise. Note that we consider 5 days of the week (from Monday to Friday) and 12 h a day, from 8 o'clock in the morning to 8 o'clock in the evening.

As an example, consider the timetables depicted in Fig. 1. Let us calculate the Compactness C for the timetable (a): W_{dh} would be 1 for every day of the week, if index h takes values from 1 to 4. However, $(FR)_{dh}$ is always 0, since there are no free hours in this timetable. This way $C = 0$ for time timetable (a). Things are different if we consider timetable (b): In this case, W_{dh} is 1 for every day of the week if index h has the values from 1 to 8. In this specific example, $(FR)_{dh} = 1$ every day of the week if $h = 2$, 3, 5 and 6, i.e. 4 h a day for 5 days, $C = 20$.

References

[1] Burke EK, Ross P. The Practice and Theory of Automatic Timetabling: Selected Papers from the First International Conference, Edinburgh, 1995, Lecture Notes in Computer Science, Vol. 1153. Berlin: Springer; 1996.

[2] Burke EK, Carter MW. The Practice and Theory of Automatic Timetabling: Selected Papers from the Second International Conference, Toronto, 1997, Lecture Notes in Computer Science, Vol. 1408. Berlin: Springer; 1998.

[3] Burke EK, Carter MW. The Practice and Theory of Automatic Timetabling: Selected Papers from the Third International Conference, Konstanz, 2000, Lecture Notes in Computer Science, Vol. 2079. Berlin: Springer; 2001.

[4] Carter MW, Laporte G. Recent developments in practical course timetabling. In: Burke EK, Carter MW. The Practice and Theory of Automatic Timetabling: Selected Papers from the Second International Conference, Toronto, 1997, Lecture Notes in Computer Science, Vol. 1408. Berlin: Springer; 1998. p. 3–19.

[5] de Werra D. An introduction to timetabling. European Journal of Operational Research 1985;19:151–62.

[6] Burke EK, Petrovic S. Recent research directions in automated timetabling. European Journal of Operational Research 2002;140(2):266–80.

[7] Schaerf A. A survey of automated timetabling. Artificial Intelligence Review 1999;13:87–127.

[8] Ross P, Hart E, Corne D. Genetic algorithms and timetabling. In: Ghost A, Tsutsui S, editors. Advances in evolutionary computation, theory and applications. Berlin: Springer; 2003. p. 755–71.

[9] Carter MW. A survey of practical applications of examination timetabling algorithms. Operations Research 1986;34(2):193–201.

[10] Bardadym VA. Computer aided school and university timetabling: the new wave. In: Burke EK, Ross P, editors. The Practice and Theory of Automatic Timetabling: Selected Papers from the First International Conference, Edinburgh, 1995, Lecture Notes in Computer Science, Vol. 1153. Berlin: Springer; 1996. p. 22–45.

[11] Burke EK, Jackson KS, Kingston JH, Weare RF. Automated timetabling: the state of the art. The Computer Journal 1997;40(9):565–71.

[12] Goltz H, Kuchler G, Matzke D. Constraint-based timetabling for universities. In: Proceedings of 11th International Conference on Applications of Prolog (INAP-98) Tokyo, Japan; 1998.

[13] Gueret C, Jussien N, Boizumault P, Prins C. Building university timetables using constraint logic programming. In: Burke EK, Ross P, editors. The Practice and Theory of Automatic Timetabling: Selected Papers from the First International Conference, Edinburgh, 1995, Lecture Notes in Computer Science, Vol. 1153. Berlin: Springer; 1996. p. 130–45.

[14] White GM. Constrained satisfaction, not so constrained satisfaction and the timetabling problem. In: Burke EK, Carter MW, editors. The Practice and Theory of Automatic Timetabling: Selected Papers from the Third International Conference, Konstanz, 2000, Lecture Notes in Computer Science, Vol. 2079. Berlin: Springer; 2001. p. 32–47.

[15] Abdennadher S, Marte M. University timetabling using constraint handling rules. In: Proceedings of Journees Francophones de Programmation Logique et Programmation par Constraintes; 1998.

[16] Mooney EL, Rardin RL, Parmenter WJ. Large scale classroom scheduling. IIE Transactions 1996;28(5):369–78.

[17] Colorni A, Dorigo M, Maniezzo V. Metaheuristics for high-school timetabling. Computational Optimization and Applications 1998;9(3):275–98.

[18] Burke EK, Elliman DG, Weare RF. A hybrid genetic algorithm for highly constrained timetabling problems. In: Proceedings of the Sixth International Conference on Genetic Algorithms. San Francisco, CA: Morgan Kaufmann; 1995. p. 605–10.

[19] Abramson D. Constructing school timetables using simulated annealing: sequential and parallel algorithms. Management Science 1991;37(1):98–113.
[20] Monfroglio A. Hybrid genetic algorithms for timetabling. International Journal of Intelligent Systems 1996;11(8): 477–523.
[21] Monfroglio A. Timetabling through constrained heuristic search and genetic algorithms. Software—Practice and Experience 1996;26(3):251–79.
[22] Rudová H, Matyska L. Timetabling with annotations. Technical report FIMU-RS-99-09, Faculty of Informatics Masaryk University; 1999.
[23] Rudová H, Matyska L. Constraint-based timetabling with student schedules. In: Burke EK, Carter MW, editors. The Practice and Theory of Automatic Timetabling: Selected Papers from the Third International Conference, Konstanz, 2000, Lecture Notes in Computer Science, Vol. 2079. Berlin: Springer; 2001. p. 103–23.
[24] Rudová H, Murray K. Universitary course timetabling with soft constraints. In: Burke EK, Carter MW, editors. The Practice and Theory of Automatic Timetabling: Selected Papers from the Fourth International Conference, Gent, 2002, Lecture Notes in Computer Science, Vol. 2740. Berlin: Springer; 2003. p. 73–89.
[25] Paechter B, Rankin RC, Cumming A, Fogarty TC. Timetabling the classes of an entire university with an evolutionary algorithm. In: Proceedings of Fifth International Conference on Parallel Problem Solving from Nature Amsterdam, The Netherlands; 1998.
[26] Fang H, Ross P, Corne D. A promising hybrid GA/Heuristic approach for open-shop scheduling problems. In: Cohn A, editor. Proceedings of 11th European Conference on Artificial Intelligence. New York: Wiley; 1994. p. 590–4.
[27] Lai WK, Coghill GC. Channel assignment through evolutionary optimization. IEEE Transactions on Vehicular Technology IEEE Press, Piscataway, NJ 1996;55(1):91–5.
[28] Michalewicz Z. Genetic algorithms + data structures = evolution programs. Berlin: Springer; 1992.
[29] Bäck T. Selective pressure in evolutionary algorithms: a characterization of selection mechanisms. In: Proceedings of First IEEE Conference on Evolutionary Computation; 1994.
[30] Dasgupta D, Michalewicz Z, editors. Evolutionary algorithms in engineering applications. Berlin: Springer; 1997.
[31] Goldberg DE. Genetic algorithm in search, optimization and machine learning. Reading, MA: Addison-Wesley; 1989.
[32] Stamatopoulos P, Viglas E, Karaboyas S. Nearly optimum timetable construction through CLP and intelligent search. International Journal on Artificial Intelligence Tools 1998;7(4):415–42.
[33] Smith KA, Abramson D, Duke D. Hopfield neural networks for timetabling: formulations, methods, and comparative results. International Journal of Computers and Industrial Engineering 2003;44(2):283–305.