Available online at www.sciencedirect.com

ScienceDirect

Computers & Operations Research 35 (2008) 1265 – 1280

computers & operations research

www.elsevier.com/locate/cor

Applying evolutionary computation to the school timetabling problem: The Greek case

Grigorios N. Beligiannis[a, b, *], Charalampos N. Moschopoulos[a], Georgios P. Kaperonis[a], Spiridon D. Likothanassis[a]

[a] *Department of Computer Engineering and Informatics, University of Patras, GR-26500, Rio, Patras, Greece*
[b] *Department of Farm Organization and Management, School of Natural Resources and Enterprise Management, University of Ioannina, G. Seferi 2, GR-30100, Agrinio, Greece*

Available online 10 October 2006

Abstract

In this contribution, an adaptive algorithm based on evolutionary computation techniques is designed, developed and applied to the timetabling problem of educational organizations. Specifically, the proposed algorithm has been used in order to create feasible and efficient timetables for high schools in Greece. The algorithm has been tested exhaustively with real-world input data coming from many different high schools and has been compared with several other effective techniques in order to demonstrate its efficiency and superior performance. Simulation results showed that the algorithm is able to construct a feasible and very efficient timetable more quickly and easily compared to other techniques, thus preventing disagreements and arguments among teachers and assisting each school to operate with its full resources from the beginning of the academic year. Except from that, due to its inherent adaptive behavior it can be used each time satisfying different specific constraints, in order to lead to timetables, thus meeting the different needs that each school may have.

Keywords: Evolutionary algorithms; School timetabling problem; Educational organizations; Computer application; Artificial intelligence

1. Introduction

The problem faced in this contribution belongs to the wide category of timetabling problems. These problems deal with the allocation of resources to specific time periods so that some specific constraints are satisfied and the created timetable is feasible and effective. According to each case, the constraints, the sources and the elements defining the effectiveness of the timetable are determined. These timetabling problems are NP-complete in their general form, regarding their computational complexity, meaning that the difficulty to find a solution rises exponentially to their size and a deterministic algorithm, giving an acceptable solution in polynomial time, cannot be found. Therefore, alternative optimization methods have been used in order to reach a (near) optimal solution [1–8]. Other methods used in order to solve the timetabling problem are local search techniques [9,10], case-based reasoning [11] and heuristic orderings [12].

* Corresponding author. Department of Computer Engineering and Informatics, University of Patras, GR-26500, Rio, Patras, Greece. Tel.: +30 2610 996985; fax: +30 2610 969001.

E-mail addresses: beligian@ceid.upatras.gr (G.N. Beligiannis), mosxopul@ceid.upatras.gr (C.N. Moschopoulos), kaperoni@ceid.upatras.gr (G.P. Kaperonis), likothan@cti.gr (S.D. Likothanassis).

0305-0548/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2006.08.010

Lately, many computational intelligence algorithms and techniques including genetic algorithms [13,14], evolutionary [15,16], tabu search [17,18], ant systems [19] and metaheuristics [20] have been applied in order to demonstrate their effectiveness and reliability concerning this major problem category.

The timetabling problem for educational organizations constitutes a subcategory of the above-mentioned categories of timetabling problems. It usually refers to exams or lessons timetabling at a school or a university [21,22]. Wren [23] defines timetabling as a special case of scheduling: "Timetabling is the allocation, subject to constraints, of given resources to objects being placed in spacetime, in such a way as to satisfy as nearly as possible a set of desirable objectives". In a more common definition, timetabling problem consists in fixing in time and space, a sequence of meetings between teachers and students, in a prefixed period of time, satisfying a set of constraints of several different kinds. These constraints may include both hard constraints that must be respected and soft constraints used to evaluate the solution quality. [24,25].

This contribution is focused on the problem of constructing a feasible and efficient timetable for high schools in Greece [26,27]. The situation which holds nowadays in Greece is the following: at the beginning of each academic year, the teachers of each school are quickly making a temporary and incomplete timetable and during the next one or two months they are trying to improve it and produce its final complete version. So, for a quite long period of time the school is not working with its full resources. Apart from that, every teacher understands in his/her own way the term "quality" regarding a timetable. As a result, many disagreements and arguments, concerning the final version of the timetable, occur among the teachers. To avoid this problematic situation we propose the use of an evolutionary algorithm (EA) in order to create automated and more "objective" timetables which can be more easily accepted by all teachers. Such an algorithm can construct a "quality" timetable in only few minutes and in this case the "quality" of the timetable is formally defined. As a result, arguments, between the teachers who construct the timetable and the rest ones, are prevented. So, by using computers and an automated timetabling constructing algorithm, "quality" timetables can quickly and easily be created.

In this contribution, an adaptive algorithm based on evolutionary computation techniques has been designed, developed and applied to the timetabling problem of educational organizations in Greece. Specifically, the algorithm proposed has been used in order to create feasible and efficient timetables for Greek high schools. The algorithm has been tested exhaustively with real-world input data coming from many different Greek high schools in order to demonstrate its quality and efficiency. Simulation results showed that the algorithm is able to construct a feasible and very efficient timetable more quickly and easily compared to other techniques applied to the same problem. In order to be more convincing about the effectiveness and efficiency of the proposed algorithm we compare its performance with two other algorithms which have been applied to the Greek timetabling problem with excellent results [26,27]. Simulation results showed that the proposed EA achieves better results compared to these two very effective algorithms, while requiring less execution time. However, the main advantage of the proposed algorithm lies in its adaptive behavior. By assigning weights, that can be defined by the user, to each specific constraint that should be satisfied, it allows teachers to guide the algorithm to solutions that will better fulfill their specific needs. So, the algorithm can be used each time in order to result in timetables satisfying different specific constraints, thus meeting the different needs that each school may have.

The rest of the paper is organized as follows: Section 2 defines the high-school timetabling problem in Greek high schools. Section 3 introduces the proposed EA relative to chromosome encoding, initialization procedure, fitness function, hard and soft constraints, cost assignment, genetic operators (selection, crossover and mutation) and elitism schema. Section 4 presents the experimental results and the performances obtained. This section also includes a useful discussion concerning these performances relative to some important aspects of the proposed algorithm. In Section 5 some interesting conclusions are drawn.

2. Problem definition

The weekly school timetable is affected by many parameters and must satisfy a large number of constraints [15,28–32]. The entities that are involved in such a timetable are the teachers, the lessons, the classes and the time periods. In particular, the teachers teach some lessons at the classes and this process takes place in specific time periods. Therefore, constructing a timetable means to assign, for the entire teacher–lesson–class triples, the time periods that the teaching process will take place. This has to be done in a way that as many as possible constraints are satisfied, some of them partially and some of them totally. These constraints can be divided into two groups: "hard" constraints

	Monday	Tuesday	Wednesday	Thursday	Friday
1st Hour	1	8	15	22	29
2nd Hour	2	9	16	23	30
3rd Hour	3	10	17	24	31
4th Hour	4	11	18	25	32
5th Hour	5	12	19	26	33
6th Hour	6	13	20	27	34
7th Hour	7	14	21	28	35

Fig. 1. Relation between days–hours–time periods.

and "soft" constraints. Both of them can be also divided into the ones related to teachers and the ones that are related to classes. When all hard constraints are satisfied then a "feasible" timetable is constructed, meaning a timetable that can actually be used by the school it was made for. However, as it will be presented in the next paragraphs, the quality of a timetable is mainly influenced by the proportion of soft constraints satisfied. The basic aim is to construct a feasible timetable while maximizing its quality. The formal way in which "quality" is measured will be discussed in the next subsection.

In order for the construction of the timetable to be started all the necessary input data must be available. These data will provide all the necessary information about the relations between teachers, classes and lessons. This information comprises the lessons taught by each teacher, the hours taught by each teacher per week as well as the classes to which these lessons are taught. Moreover, information about the co-teaching and subclasses cases should be available. Finally, important information is also the days each teacher is available at school. All the above information should be given as input data. The output is, of course, the weekly timetable, which shows for every teacher the class he/she is assigned to it at every hour in a week. From this timetable someone can easily derive all conclusions that are related to its quality (meaning how many and which constraints are violated).

It should be noted at this point that a weekly school timetable in Greece is divided typically into five days and each one of them into seven hours. The term "time period" refers to each hour of teaching in a week and consequently the number of each time period defines the day and the hour it takes place. The number of time periods for a typical Greek high school is shown in Fig. 1.

Hard constraints that must be satisfied in order to keep the timetable feasible are the following:

- Each teacher must not be assigned to more than one class in a specific time period.
- Each teacher must teach only to classes that are initially assigned to him/her and only lessons that he/she is supposed to teach to these classes.
- Each teacher must teach a limited and defined number of hours and lesson(s) to each class.
- Each teacher must only teach at days he/she is available at school.
- Each class must not have more than one lesson in a specific time period.
- The number of lessons and hours each class must be taught by specific teachers are determined by input data.
- The classes' empty (no teacher assigned) time periods, whose total acceptable number is determined by input data implicitly, must be the last hour of each day.

Soft constraints that should be satisfied in order the timetable to be considered of high quality are the following:

- Teachers should have as few empty periods as possible.
- The total empty periods existing should be uniformly distributed to all teachers while the empty periods of each teacher should be uniformly distributed to all days he/she is available at school.
- The number of hours a teacher has lessons each day should not differ dramatically among these days.
- Classes should not have the same lesson in successive hours and, if possible, not even during the same day.

Apart from the above, there are some extra factors that affect the structure of a timetable. These factors are related to co-teaching and subclasses cases. During the last decade it is common in many schools that some classes are divided, only for the teaching of some specific lessons, into two temporary ones. This is the case especially for lessons like "foreign language" where every class is divided into two temporary ones according to the level of students' knowledge ("advanced" class and "beginners" class). In such cases, the teachers who teach the specific lesson to the temporarily divided class must be assigned to this class at the same time period. This case is referred to as co-teaching. The subclasses case refers to the circumstance where a class is divided into two temporary subclasses in which different lessons are taught. This is the case especially when lab lessons, like "informatics" and "technology", are concerned. In such cases, the teachers who teach these different lessons to the subclasses of the temporarily divided class must be assigned to this class at the same time period. In our approach, both co-teaching and subclasses cases are considered as hard constraints.

All the above affect the construction process and quality of a timetable. Having all these in mind, an algorithm for school timetabling generation has been designed, which is presented in Section 3.

2.1. Mathematical model

The necessary parameters and data sets needed for the problem's model definition are the following:

- *T* is the set of teachers.
- *C* is the set of classes.
- *D* is the set of days in the timetable.
- *H* is the set of teaching hours.
- *L* is the set of lessons.
- *Teacher_Class_Hour_Day[t][c][h][d]* is defining if teacher *t* is assigned to teach class *c* at hour *h* in day *d*.
- *Class_Lesson_Hour_Day[c][l][h][d]* is defining if at class *c* lesson *l* is taught at hour *h* in day *d*. If *Class_Lesson_Hour_Day*[*c*][*l*][*h*][*d*] equals 1 this means that class *c* is taught lesson *l* at hour *h* in day *d*, otherwise if *Class_Lesson_Hour_Day[c][l][h][d]* equals 0 this means that class *c* is taught no lesson (empty period) at hour *h* in day *d*.
- *Teacher_Total_Hours[t]* is the total teaching hours assigned to teacher $t \in T$.
- *Class_Total_Hours[c]* is the total teaching hours assigned to class $c \in C$.

Except for that, the following costs are defined:

- $cost_1$: The empty periods of each teacher (they must be as few as possible).
- $cost_2$: The empty periods of each teacher not uniformly distributed (they should be uniformly distributed to all days he/she is available at school).
- $cost_3$: The total empty periods of all teachers not uniformly distributed (they should be uniformly distributed to all teachers).
- $cost_4$: The number of hours a teacher has lessons not uniformly distributed among the days he is available at school (they should be uniformly distributed among these days).
- $cost_5$: Classes having the same lesson in successive hours or in the same day (they should not have the same lesson in successive hours or in the same day).

So, the mathematical model of the problem can be expressed as follows:

$$\min(cost_1 + cost_2 + cost_3 + cost_4 + cost_5) \quad \forall t \in T,\ \ c \in C,\ \ l \in L,\ \ h \in H,\ \ d \in D, \tag{1}$$

under the following constraints:

1. $\forall t \in T, c_i, c_j \in C, h_i, h_j \in H, d_i, d_j \in D \notin$ (*Teacher_Class_Hour_Day[t]*$[c_i][h_i][d_i]$, *Teacher_Class_Hour_Day[t]*$[c_j][h_j][d_j]$) such that $c_i \neq c_j$ and $h_i = h_j$ and $d_i = d_j$.
2. $\forall c \in C, l_i, l_j \in L, h_i, h_j \in H, d_i, d_j \in D \notin$ (*Class_Lesson_Hour_Day[c]*$[l_i][h_i][d_i]$, *Class_Lesson_Hour_Day[c]*$[l_j][h_j][d_j]$) such that $l_i \neq l_j$ and $h_i = h_j$ and $d_i = d_j$.

3. $\forall t \in T$, *Teacher_Total_Hours[t]* $=$ *Teacher_Hours[t]*, where *Teacher_Hours[t]* is the total teaching hours that each teacher can teach according to law.
4. $\forall c \in C$, *Class_Total_Hours[c]* $=$ *Class_Hours[c]*, where *Class_Hours[c]* is the total teaching hours that can be taught to each class according to law.
5. $\forall c \in C, l \in L, h \in H, d \in D \notin$ $Class\_Lesson\_Hour\_Day[c][l][h][d] = 0$ such that h is not the last teaching hour of a day.
6. $\forall t \in T, c \in C, h \in H, d \in D \notin$ *Teacher_Class_Hour_Day[t][c][h][d]* such that d is a day that teacher t is not available at school.
7. $\forall t_i, t_j \in T, c_i, c_j \in C, h_i, h_j \in H, d_i, d_j \in D, \notin$ $Teacher\_Class\_Hour\_Day[t_i][c_i][h_i][d_i]$, $Teacher\_Class\_Hour\_Day[t_j][c_j][h_j][d_j]$ such that $c_i = c_j$ and $h_i = h_j$ and $d_i = d_j$ and teachers t_i and t_j are not teaching the same lesson to class $c_i = c_j$ (case of co-teaching).

3. The proposed EA

In order to solve the Greek school timetabling problem, an automated timetable construction EA has been designed and implemented. The motivation that led us to design, implement and apply an EA to this specific problem is that computational intelligence techniques have been already widely used for solving the school timetabling problem with very satisfactory results [33–36]. A first attempt to apply a genetic algorithm on the Greek school timetabling problem is presented in [37]. The structure and operation of the proposed algorithm is described in the following subsections.

3.1. Chromosome encoding

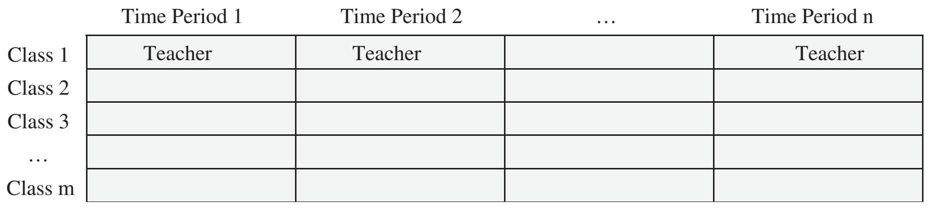
The chromosome encoding used is one of the most significant factors in order for the EA to converge satisfactorily and fast to an (near) optimal solution. The term "encoding" is used to describe a way of representing a weekly timetable (feasible or not). Namely, every chromosome must encode all the information needed for the description of a timetable. The encoding designed for this algorithm is quite different from others already proposed in the literature [15,28,29,31,38]. The selection of the chromosome encoding was made in order to satisfy the following criteria:

- To occupy as small space as possible in relevance with the information it encodes.
- To be easy to preserve, during the reproduction processes, as much as possible characteristics of the timetable concerning hard constraints.

At first, we decided not to include the entity lessons directly in the chromosome encoding. This is because lessons are a factor which increases dramatically the search space, by increasing the dimension of the problem, while it does not play a decisive role in order to construct an (near) optimal timetable. For example, if a teacher is assigned to teach two hours "literature", three hours "ancient Greek" and one hour "history" to specific class, useful information consists of the specific hours that the teacher is assigned to the class and not of the lessons he is actually teaching during these hours. Information about lessons is only needed in order to avoid cases where a teacher is assigned to teach the same lesson in a class for two or more hours in a row, which is not permitted. More analytically, a teacher who teaches two different lessons to a specific class can be assigned to teach two hours in a day to this class, while this is not permitted for a teacher who teaches only one lesson to this class. So, it is obvious that the only information needed concerning lessons is not the lessons themselves but only the number of different lessons each teacher is teaching to each class. This factor is included in the chromosome encoding.

A Chromosome is represented by a two-dimensional array $m \times n$, where m is the number of classes and n is the number of time periods per class in a week. This means that if the weekly timetable is divided into five days and each day can have at most seven teaching hours per class (this is typical for Greek high schools) the number of cells of the array equals 35. In every cell a number is stored, representing a teacher assigned to specific class at specific time period. This chromosome encoding is shown in Fig. 2. The position of each cell shows implicitly to which class and time period each teacher is assigned.

The above chromosome encoding serves satisfactorily all the roles it is designed for. In particular, the array uses the minimum space for the encoding since no cell hosts unnecessary information and though it is a dense array. Apart from that, it encodes all the necessary information such as the initial assignment of teachers and hours of lessons to



	Time Period 1	Time Period 2	…	Time Period n
Class 1	Teacher	Teacher		Teacher
Class 2				
Class 3				
…				
Class m				

Fig. 2. Chromosome encoding.

classes. This means that every chromosome (and for every generation) encodes a timetable having no mistakes in the relationships between teachers and classes (a teacher cannot be assigned to a class he/she is not supposed to teach). Moreover, the total weekly teaching hours of each teacher are always right according to the initial distribution. Finally, the subclasses case is handled by assigning only one of the teachers to the class and assuming that when this is the case, the other teacher should also be treated as been assigned to this class. So, there is no need to check for violations in the assignment of teachers to subclasses. The above characteristics are the ones each chromosome must preserve during the evolution process in order to be a feasible one. This is the case in the proposed EA since all genetic operators are designed so as not to violate these rules.

3.2. Initialization procedure

The initialization procedure is an important issue in every EA because it should create a random population with as much diversity as possible. It is desirable that the individuals that constitute the first generation are spread in the whole search space and represent it as well as possible. This gives the algorithm the opportunity to search effectively the whole space of possible solutions and not to be trapped prematurely in local optima. In the specific EA presented, the initialization process is even more important due to the fact that it is also the process that encodes the input information into the chromosome's representation. The initialization procedure consists of the following steps:

```
For each chromosome do
  For each class do
    For each time period do
      1. Choose a teacher at random.
      2. If the selected teacher has not been assigned all the teaching hours he has to teach, then assign to this
         teacher the selected class for the selected time period. Otherwise go to step 1.
      3. Decrease the "not assigned teaching hours" of this teacher—class pair by one.
      4. If the teacher–class pair defines a co-teaching case or a subclasses case, do the corresponding assignments
         as defined by input data.
    end
  end
end.
```

3.3. Fitness function

The fitness function is the most important part of the EA. It decodes the chromosome into a timetable and returns a value representing the fitness of each specific chromosome, that is, how well it solves the problem faced. In addition, this is the function which checks if hard constraints and most of soft constraints are satisfied by each chromosome (school timetable). According to the constraint violation check done, every constraint violation is assigned a subcost and the sum of all subcosts having to do with a specific chromosome is the total cost of this chromosome.

Analytically, the following hard and soft constraints are considered:

3.3.1. Hard constraints

The first hard constraint checked concerns teachers who are assigned to more than one class in a time period. The number of time periods that are not feasible for a teacher and the number of teachers violating this constraint are the factors affecting the cost of this specific constraint.

The second hard constraint checked concerns teachers who are assigned to a class in a specific time period but they are not available at school in that day. The number of teachers violating this constraint affects its cost. Information about teachers' availability at school is not encoded in the chromosome but in a different structure that is generated during the initialization procedure.

The third hard constraint checked concerns classes which should be assigned to the same teacher in a specific time period, due to co-teaching, but are not. The number of classes violating its constraint affects its cost. Information about co-teaching is not encoded in the chromosome but in a different structure, as well.

The fourth hard constraint concerns classes having an empty period at a day which is not the last hour of this day. The number of these empty periods affects the respective cost.

3.3.2. *Soft constraints*

The first soft constraint checked concerns teachers having empty periods in their timetable. The cost of this constraint is affected by the number of empty periods, the number of days they occur and, of course, the number of empty periods per day.

The second soft constraint checked concerns the dispersion of teaching hours in the teachers' timetable among working days. The objective is that teaching hours should be uniformly distributed among the days. The ideal case for a teachers' timetable is the one that assigns to every Teacher $\lfloor m \rfloor$ or $\lceil m \rceil$ teaching hours in every day, where m is the average teaching hours per day for each teacher (i.e. the total teaching hours per week divided by the number of days the teacher is available at school). Every declination from the ideal dispersion affects the cost of this constraint.

The third soft constraint concerns the dispersion of lessons in the classes' timetable. The ideal case is the one in which every class has as many different lessons as possible in every day (no matter if they are taught by the same teacher or not). Every declination from the ideal dispersion affects the cost of this constraint, as well.

3.3.3. *Cost assignment*

The total cost of each chromosome evaluated equals the weighted sum of the subcosts corresponding to each type of constraint violation. Apart from the total cost of each chromosome, the program also stores the cost of every teacher's timetable for every chromosome. This value is useful to the mutation operator as we will discuss later. The weights used in order to calculate the (weighted sum) cost of each genome (timetable) are the following:

1. *Hard constraints weight* (*HCW*): This weight assists the EA to distinguish feasible and infeasible timetables. This weight should have a much greater value compared to the values of other weights, so that a feasible but "not efficient" timetable will rarely have a bigger cost compared to an infeasible timetable. At the same time, however, this weight's value should not be extremely big in order not to eliminate the differences between infeasible timetables.
2. *Teacher empty periods weight* (*TEPW*): This weight has to do with the empty hours in a teacher's timetable. As it has already been stated, empty hours in a teacher's timetable are the ones that mainly determine its quality.
3. *Ideal dispersion weight for teachers* (*IDWT*): This weight has to do with the dispersion of teaching hours in a teachers' timetable.
4. *Ideal dispersion weight for classes* (*IDWC*): This weight has to do with the dispersion of lesson hours in a classes' timetable.

Another major parameter that affects the behavior of the evaluation function is the exponential rise base (BASE). This is a real number (typically between 1 and 2) which is used as a base for the exponential rise of the subcosts corresponding to violations of a certain constraint in a timetable. The value of this parameter can be set by the user according to his/her needs. For example, if we would like two empty hours existing in the timetable of the same teacher to have greater cost than two empty hours each existing in the timetable of a different teacher, the value of parameter BASE must be greater than 1. In this way, every additional empty space in the timetable of the same teacher will increase more the cost of the corresponding subcost, compared to the cost of each first empty space. By using this technique, timetables with high costs are rejected.

Next, the basic subcosts, used to evaluate the total cost of each chromosome, are presented. The total cost of each timetable is calculated by adding the subcosts corresponding to each of the following cases:

1. *Teacher unavailability*: For each teacher and for each time period this teacher is assigned to a class, while he/she is unavailable at school the corresponding day, the following cost is added: $HCW * BASE^3$.

2. *Teacher assigned in more than one class*: For each teacher and for each time period, if this teacher is assigned to more than one class, for example, to k classes, the following cost is added: $HCW * BASE^k$.
3. *Class's empty spaces*: For each class and for each time period that no lesson is assigned to this class, if this time period is not the last of the corresponding day, then the following cost is added: $HCW * BASE^{\mathrm{BASE}}$.
4. *Wrong co-teaching*: For each class and for each time period that a teacher is assigned to this class implying co-teaching, if the other teacher is not assigned to the same class at the same time period, the following cost is added: $HCW * BASE^{\mathrm{BASE}}$.
5. *Teacher's empty spaces*: For each teacher the following subcost is added: $TEPW * HOURS * BASE^{\mathrm{DAYS}}$, where *HOURS* is the total number of empty time periods in the teacher's timetable and *DAYS* is the number of days that empty time periods exist in the timetable of this specific teacher.
6. *Teacher lessons' dispersion*: For each teacher the following subcost is added: $IDWT * HOURS * BASE^{\mathrm{DAYS}}$, where *HOURS* is the total number of time periods that are different compared to the ideal dispersion teacher's timetable and *DAYS* is the number of days in the teacher's timetable that such a declination from the ideal case exists.
7. *Class lessons' dispersion*: For each class the following subcost is added: $IDWC * HOURS * BASE^{\mathrm{DAYS}}$, where *HOURS* is the total number of time periods that the same lessons are assigned to the same day, and *DAYS* is the number of days in the class's timetable that this case occurs.

So, the mathematical representation of the evaluation function used, which was presented in parts above, is the following:

$$\begin{aligned}&\min[\mathrm{cost(teacher\_unavailability)} + \mathrm{cost(teacher\_assigned\_in\_more\_than\_one\_class)}\\&\quad + \mathrm{cost(class'\ s\_empty\_spaces)} + \mathrm{cost(wrong\_co\text{-}teaching)} + \mathrm{cost(teacher'\ s\_empty\_spaces)}\\&\quad + \mathrm{cost(teacher\_lessons'\_dispersion)} + \mathrm{cost(class\_lessons'\_dispersion)}]\end{aligned} \tag{2}$$

or

$$\begin{aligned}&\min\,[\mathrm{cases\_of\_teacher\_unavailability} * HCW * BASE^3\\&\quad + \mathrm{cases\_of\_teacher\_assigned\_in\_more\_than\_one\_class} * HCW * BASE^k\\&\quad + \mathrm{cases\_of\_\ class'\ s\_empty\_spaces} * HCW * BASE^{\mathrm{BASE}}\\&\quad + \mathrm{cases\_of\_wrong\_co\text{-}teaching} * HCW * BASE^{\mathrm{BASE}}\\&\quad + \mathrm{cases\_of\_teacher'\ s\_empty\_spaces} * TEPW * HOURS * BASE^{\mathrm{DAYS}}\\&\quad + \mathrm{cases\_of\_\ teacher\_lessons'\_dispersion} * IDWT * HOURS * BASE^{\mathrm{DAYS}}\\&\quad + \mathrm{class\_lessons'\_dispersion} * IDWC * HOURS * BASE^{\mathrm{DAYS}}].\end{aligned} \tag{3}$$

The respective weights are described in detail above.

3.4. Genetic operators

The genetic operators used in the proposed EA are presented in the following subsections. Surprising is the fact that, after experimenting a lot with many different cases of high schools, we decided not to use a crossover operator. That is because experimental results have shown that crossover in this specific problem and chromosome encoding does not contribute satisfactorily, while it adds too much complexity and time delay.

3.4.1. Selection operator

The selection operator is the one that after the evaluation of the chromosomes is completed, chooses some of them in order to create the intermediate population (mating pool) for reproduction [39–41]. After carrying out many experiments we decided to use linear ranking selection rather than the "classic" Roulette wheel selection [39,40,42], since, according to the experiments, the first one had much better results compared to the second one. Roulette wheel selection is directly affected by the cost values that the evaluation function assigns to each chromosome. To be more specific, the differences in the values of these costs are the ones that affect the selection probability of the respective chromosomes. Chromosomes representing infeasible timetables are assigned a very big cost and as a result they have a nearly zero probability of being selected. So, when a chromosome representing a feasible timetable is found all the mating pools end up consisting only of copies of this specific chromosome. This of course leads to premature convergence and not optimal timetables. Linear ranking selection, on the other hand, is not affected by the cost values themselves, but only by the classification of the chromosomes according to these values. The linear ranking selection algorithm used can be summarized as follows:

1. Sort the chromosomes of the current population according to their cost values in descending order.
2. Assign to each chromosome the value estimated according to the following formula: $p_i = (1/N)(n^- + (n^+ - n^-)(i-1)/(N-1))$, where N is the population size, i is the ith worse chromosome according to the evaluation function, n^+ equals $2 - n^-$ and n^- equals 0.2.
3. Repeat step 4 for as many times as the population size.
4. Create a random number between zero (included) and one (excluded). Find (starting from the chromosome with the smaller cost) the first chromosome such that its cost value is bigger than the value estimated from the multiplication of the cost value of the worse chromosome times this random number.

3.4.2. Mutation operator

Our goal was to create a mutation operator [39,40,42,43] which, based on chromosome encoding, will preserve all the necessary input information which is encoded in every individual. The first mutation operator designed and implemented is the "period mutation" operator. It is applied to the timetable of a class, that is, to a row of the array representing each chromosome in the selected encoding, and swaps the teachers between two time periods. As someone can easily see, this does not affect the existing allocation of teachers to classes, since both teachers swapped had a lesson to this specific class. The change relates only to the time periods that these teachers were assigned to teach their lessons. Another mutation operator used is "bad period mutation" which, in combination with "period mutation" improved the experimental results. "Bad period mutation" works as follows: the time periods chosen for swapping are not selected at random, as in "period mutation". They are the two most costly time periods regarding the timetable of teachers assigned to these time periods. The mutation operators used were designed in such a way that they provoke no violation concerning the availability of teachers and the co-teaching cases.

The two mutation operators described above have been merged into one single mutation procedure which is described in the following:

1. For each chromosome of the current population run the following steps.
2. For each class of each chromosome run the steps below.
3. Create a random number between 0 and 1.
4. If this random number is less than the "period mutation" probability, then choose at random two different time periods and swap the corresponding teachers.
5. Create another random number between 0 and 1.
6. If this random number is less than the "bad mutation" probability, then choose two different time periods so that the corresponding teachers are the ones with the worst teacher timetable and swap these teachers.

In Fig. 3, the application of the mutation operator used, is presented. It is applied to "Class i" and between time periods "2" and "n". In the case of "period mutation" the time periods are selected at random, while in the case of "bad mutation" they are selected because the "teacher A" and "teacher B" are the two teachers in this specific chromosome having the "worst" timetable.

In order the mutation operator presented above does not cause any problem concerning teachers' availability and co-teaching cases the following changes were made. After each step in which selection of two time periods to be mutated takes place (steps 4 and 6 in the above algorithm), we perform two checks in order to assure that no problems are introduced by the corresponding mutation. We first check if the mutation assigns teachers to time periods belonging to a day that they are not available at school. Only if this is not the case does the mutation take place. After that, we check if the mutation made affected a class-teacher combination corresponding to a co-teaching case. If this is the case, we make the same mutation for the class implied by the co-teaching relationship.

3.5. Elitism schema

In order to preserve the best chromosome in every generation a simple elitism schema [39,40,42] is used. The best chromosome of each generation is copied to the next generation (and not to the intermediate one, the individuals of which may mutate) assuring that it is preserved in the current population for as long as it is the best compared to the other

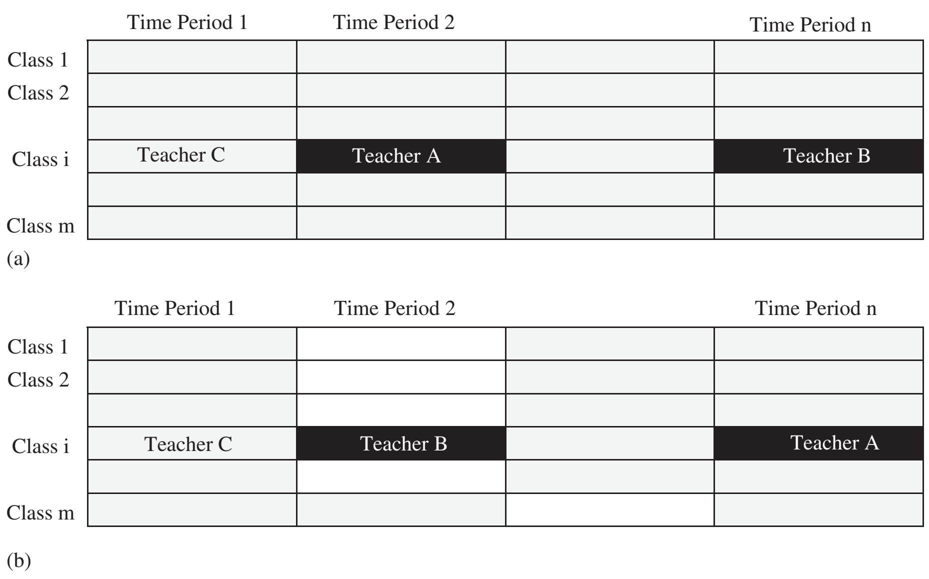


Fig. 3. (a) Before mutation and (b) after mutation.

chromosomes of the population. This choice assures that the best chromosome of each generation will be, if not better, at least equal to the best chromosome of the previous generation. Furthermore, it does not lead the EA to saturation or premature convergence since the elitism schema proposed is applied only to one chromosome of each generation.

4. Experimental results

4.1. Algorithm's performance

In order to demonstrate the efficiency and performance of the proposed algorithm, several simulation experiments were carried out. For all results presented in this section the same set of GA's operators and parameters were used in order to have a fair comparison of the algorithm's efficiency and performance. The representation used for the genomes of the genetic population is the one described in Section 3.1. As far as the reproduction operator is concerned, the linear ranking selection was used, as described in Section 3.4.1. As stated before, no crossover operator was used, while the mutation operator was the complex mutation operator presented in Section 3.4.2 (with mutation probability equal to 0.05). The size of the population was set to 25 and the algorithm was left to run for 10 000 generations. Except for that, the proposed algorithm used linear scaling [39,40] and elitism.

Firstly, the algorithm was applied to real-world input data coming from 30 different Greek high schools settled in the city of Patras. These data are available through http://prlab.ceid.upatras.gr/timetabling/. The comparison of the timetables constructed by the proposed EA and the real timetables used at schools in the city of Patras is based on three criteria. The first one is the distribution of teachers, that is, how evenly each teacher's hours are distributed among the days of the proposed timetable. For example, if a teacher has to teach 10 hours per week and is available at school for five days, the ideal "distribution of teachers" for him/her would be 2 hours of teaching per day. The second one is the distribution of lessons. One lesson should not be taught twice or more at the same day. The third one is the teachers' gaps. The teaching hours of each teacher in a day should be continuous—no idle hours are allowed to exist between teaching hours.

Regarding the distribution of teachers, two results (numbers) are presented in the respective column of Table 1. The first number is the number of teachers whose teaching hours do not have an even distribution among the days of the timetable, while the second number (between the parentheses) is the number of days that this uneven distribution occurs. For example, if a teacher is available at school for five days, has 10 teaching hours and their distribution among these five days is the following: 1-4-2-2-1, then the number of days in which uneven distribution for this teacher occurs is three. Concerning the second quantity, that is, the distribution of lessons, the first number in the respective column is the distinct number of classes in which the same lesson is taught twice or more at the same day. The second number

Table 1
Comparing timetables constructed by the proposed EA with real-world timetables used at schools

Test Data set	Number of teachers	Number of classes	Teaching hours	Timetables used at schools			Timetables created by the EA			
				Distribution of teachers	Distribution of lessons	Teacher's gaps	Distribution of teachers	Distribution of lessons	Teacher's gaps	Time (min)
1	34	11	35	21(48)	1(1)	25(34)	18(40)	1(1)	25(29)	30
2	35	11	35	15(34)	0(0)	30(52)	15(34)	0(0)	26(42)	30
3	19	6	35	9(23)	3(3)	8(24)	4(8)	3(3)	9(9)	24
4	19	7	35	6(14)	0(0)	15(31)	5(12)	0(0)	17(29)	24
5	18	6	35	6(17)	0(0)	15(39)	1(2)	0(0)	8(8)	22
6	34	10	35	24(47)	10(25)	9(9)	18(33)	10(20)	7(7)	30
7	35	13	35	24(56)	13(36)	24(33)	24(50)	13(27)	24(32)	45

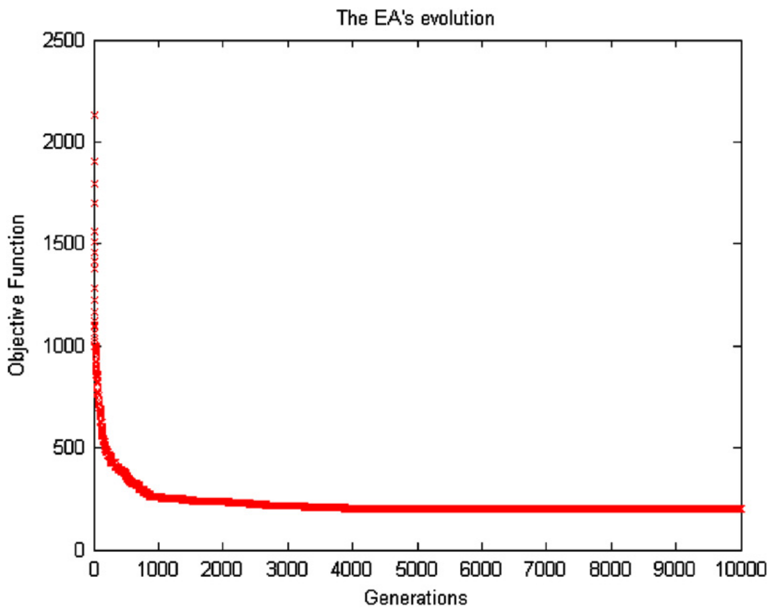


Fig. 4. The evolution of the proposed algorithm for test data set 2 (Table 1).

(between the parentheses) is the total number of classes in which this incident occurs per week. A class is counted more than once in estimating this number, if there is more than one day that the same lesson is taught twice or more in this class. Finally, regarding the teachers' gaps, the first number in the respective column is the distinct number of teachers in whose timetable idle hours exist, while the second number (between the parentheses) is the total number of idle hours for all teachers.

In Table 1 the performance and efficiency of the proposed EA is shown by comparing the timetables constructed by it with the real timetables used at schools in the city of Patras. These data was given to us by the Headship of Second Grade Education of Achaia county (http://www.dide.ach.sch.gr/). A more detailed description of this data can be found through http://prlab.ceid.upatras.gr/timetabling/. In the second, the third and the fourth column of Table 1 the number of teachers, the number of classes and the number of teaching hours of each specific school are presented, respectively. These parameters define the size of each school and affect the size and the complexity of the respective optimal timetable.

In Figs. 4 and 5, we present the evolution of the proposed EA in order to demonstrate its convergence. Our aim is to show that although crossover is not included in the algorithm, mutation is enough to generate new good solutions to evolve the population.

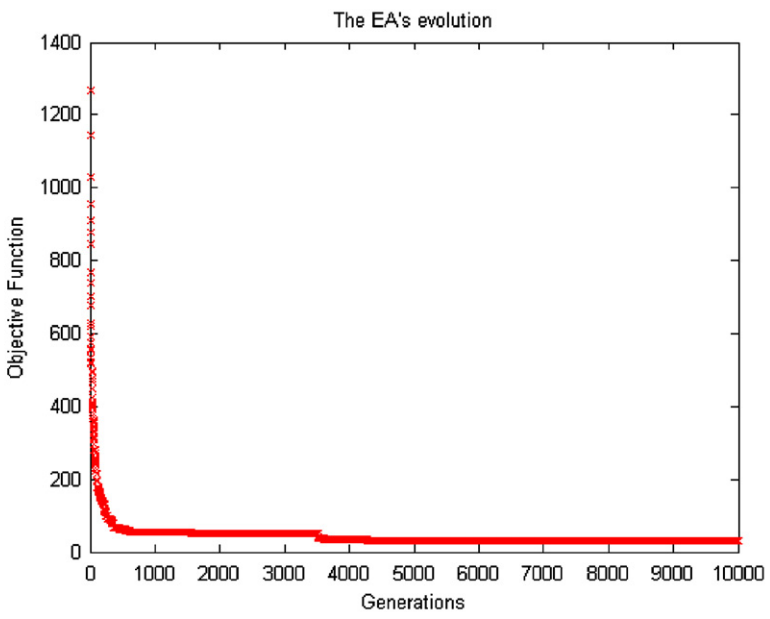


Fig. 5. The evolution of the proposed algorithm for test data set 5 (Table 1).

Table 2
Comparing timetables constructed by the proposed EA with timetables constructed by the column generation approach presented in [26]

Test data set	Number of teachers	Number of classes	Teaching hours	Timetables created in [26]				Timetables created by the EA			
				Distribution of teachers	Distribution of lessons	Teacher's gaps	Time (min)	Distribution of teachers	Distribution of lessons	Teacher's gaps	Time (min)
1	11	5	30	5(10)	5(15)	0(0)	Less than 1 h	3(6)	5(15)	0(0)	22
2	15	6	35	6(12)	6(22)	0(0)	Less than 1 h	7(14)	6(21)	0(0)	24
3	17	7	30	6(16)	6(16)	0(0)	Less than 1 h	2(4)	7(22)	0(0)	24
4	21	9	35	6(16)	8(29)	0(0)	Less than 1 h	6(13)	9(29)	0(0)	28

Table 3
Comparing timetables constructed by the proposed EA with timetables constructed by the constraint programming approach presented in [27]

Test data set	Number of teachers	Number of classes	Teaching hours	Timetables created in [27]				Timetables created by the EA			
				Distribution of teachers	Distribution of lessons	Teacher's gaps	Time (min)	Distribution of teachers	Distribution of lessons	Teacher's gaps	Time (min)
1	11	5	30	5(10)	5(15)	0(0)	15	3(6)	5(15)	0(0)	22
2	15	6	35	5(10)	6(22)	0(0)	20	7(14)	6(21)	0(0)	24
3	17	7	30	6(16)	6(16)	0(0)	60	2(4)	7(22)	0(0)	24

In order to demonstrate its efficiency and very good performance, the proposed EA was also compared with two very effective timetabling algorithms for Greek high schools issued in the literature [26,27]. The comparison of the timetables constructed by the proposed EA and the timetables constructed by those algorithms is based not only on the three criteria mentioned above but also on the execution time. A detailed description of the high schools used in Tables 2 and 3 can be found in [26,27], respectively.

In Table 2 the performance and efficiency of the proposed EA is shown by comparing the timetables constructed by it with the timetables constructed by the column generation approach presented in [26]. The evaluation function used in this approach is the following:

$$\min \sum_{t=1}^{T} \sum_{j=1}^{COL_t} c_{t,j} x_{t,j}, \tag{4}$$

where T is the total number of teachers, $c_{i,j}$ is the cost of schedule j for teacher t, $x_{i,j}$ is a corresponding binary variable and COL_t is the number of different legal weekly schedules that have been created for teacher t. A detailed description of the above evaluation function can be found in [26].

In Table 3 the performance and efficiency of the proposed EA is shown by comparing the timetables constructed by it with the timetables constructed by the constraint programming approach presented in [27]. The objective function used in this approach is the following:

$$\min(\cos t_1 + \cos t_2). \tag{5}$$

The first term of the objective function attempts to model the desirability of the assigned teaching hours. It is calculated using the variables defining the teaching hours assigned to each teacher. For every element of these variables, a penalty is added to $cost_1$ based on whether the time requests of every teacher are satisfied. The second term of the objective function is the sum of the idle hours for all teachers. This term is minimized when all teachers have the minimum number of idle hours. A detailed description of the above objective function can be found in [27].

From the above tables, one can easily come to the conclusion that the proposed EA is very efficient and achieves better performance compared to the other two techniques applied to the same problem. Except for that in the contributions in which the other two algorithms are presented the execution time is rather bigger compared to the time needed by the proposed EA in order to be completed. As presented in the above tables the execution time of the proposed EA is less than 30 min for all cases. However, the main advantage of the proposed algorithm lies in its adaptive behavior. By assigning weights, that can be defined by the user, to each specific constraint that should be satisfied, it allows teachers to guide the algorithm to solutions that will better fulfill their specific needs. For example, if the most important factor, for a specific school, is the teachers' gaps, then we should set the respective weight to a much higher value comparatively to the other weights. So, the algorithm, by allowing teachers to set their own values to the weights in the evaluation function, can be used each time in order to result in timetables satisfying different specific constraints, thus meeting the different needs that each school may have.

4.2. Discussion

While experimenting with the EA we came to the following conclusions concerning the algorithm's performance.

4.2.1. Crossover–mutation effectiveness

The experiments made showed that the results were better if a crossover operator was not used. The use of crossover led to very slow convergence and some times even not to convergence. In our specific problem and chromosome encoding, the crossover changes totally the point where the search is taking place since it applies very big steps in the search space due to the number of changes in each chromosome. Epistasis [44] which exists as a result of our specific and "tight" chromosome representation claims that small steps in the search space should be made. This, obviously, does not happen by using a crossover operator. Therefore, it was decided not to use crossover. As far as mutation is concerned, we decided to use a complex mutation operator, comprising of two simple ones and enhance it with some necessary checks, like the reapply of the mutation process in each chromosome in which the application of the mutation operator has altered a class–teacher combination which participates in co-teaching.

4.2.2. Cost parameters

Another very important factor which dramatically affects the convergence of the algorithm is the values of the cost parameters and, in particular, the weights of constraint violations and the base of exponential rise used in the fitness function. It is obvious that the ideal values for these parameters are totally dependent on the nature of the problem. Generally, and after experimenting with various sets of parameters, we came to the conclusion that the value of the hard constraint violation should be an order of magnitude greater than the other weights. This will assure that the final timetable (the one resulted by the EA) will not in any case be an infeasible one. However, if we use an even greater value, as we have observed, the EA is driven quite soon to a feasible timetable but it is very difficult from that point to optimize the quality of the timetable, that is, the EA will not be able to minimize the soft constraints violated. So, in order to result in an as good as possible final timetable, while the population evolves, the decrease of constraints violated should be done uniformly and not partially for a specific type of constraint. On the other hand, if we assume that ideal dispersion constraint violations are not as significant as empty periods violations we should set the TESW weight to a greater value compared to the IDWT and IDWC values. Generally speaking, we came to the conclusion that there is a tradeoff between empty periods and dispersion of the teacher–class timetable. So, according to the quality demands of each school, some minor changes in the cost parameters set can—and should—be made. In every case, however, the orders of magnitude should be kept as proposed above.

4.2.3. Extension and generalization

Although the proposed algorithm is designed and developed in order to face the timetabling problem of Greek high schools, it can be easily extended in order to deal with the timetabling problem of other countries. This is because the proposed EA is inherently adaptive. The chromosome encoding used can be easily converted in order to encode all the information needed for the description of a school timetable used in another country. Except for that, the fitness function used, which is the most important part of the EA, can be easily adapted in order to include all the hard and soft constraints that have to be satisfied in order for a school timetable to be feasible in another country. Adding or extracting hard or soft constraints from the fitness function of the proposed EA can be easily done without affecting the rest of the algorithm.

5. Conclusions

In this contribution, an adaptive EA has been designed, developed and applied to the timetabling problem of Greek high schools in order to create feasible and efficient timetables. The algorithm has been tested exhaustively with real-world input data coming from many different Greek high schools in order to demonstrate its quality and efficiency. Simulation results showed that the algorithm is able to construct a feasible and very efficient timetable quickly and easily, thus preventing disagreements and arguments among teachers and assisting each school to operate with its full resources from the beginning of the academic year. The EA was also compared with two other algorithms, which have been applied to the Greek high-school timetabling problem with excellent results, in order to demonstrate its effectiveness, efficiency and superior performance. Simulation results showed that the proposed EA achieves better results compared to these two very effective algorithms, while requiring less execution time. However, the main advantage of the proposed algorithm lies in its adaptive behavior. By allowing teachers to set their own weights to each specific constraint that should be satisfied, it assists them to guide the algorithm to solutions that will better fulfill their specific needs.

References

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

[2] Kingston JH. A software architecture for timetable construction. In: Proceedings of the third international conference on practice and theory of automated timetabling, 2000. p. 472–80.

[3] Kingston JH. A tiling algorithm for high school timetabling. In: Proceedings of the fifth international conference on practice and theory of automated timetabling, 2004. p. 233–49.

[4] Abdennadher S, Marte M. University course timetabling using constraint handling rules. Applied Artificial Intelligence 2000;14(4):311–25.

[5] Trick MA. A schedule-then-break approach to sports timetabling. Lecture notes in computer science, vol. 2079. Berlin: Springer; 2001. p. 242–53.

[6] Burke E, Bykov Y, Petrovic S. A multicriteria approach to examination timetabling. Lecture notes in computer science, vol. 2079, Berlin: Springer; 2001. p. 118–31.

[7] Rudová H, Murray K. University course timetabling with soft constraints. Lecture notes in computer science, vol 2740, Berlin: Springer; 2003. p. 310–28.
[8] Dimopoulou M, Miliotis P. Implementation of a university course and examination timetabling system. European Journal of Operational Research 2001;130(1):202–13.
[9] Schaerf A, Meisels A. Solving employee timetabling problems by generalized local search. Lecture notes in computer science, vol. 1792, Berlin: Springer; 2000. p. 380–89.
[10] Burke E, Bykov Y, Newall J, Petrovic S. A time-predefined local search approach to exam timetabling problems. IIE Transactions 2004;36(6): 509–28.
[11] Burke EK, MacCarthy B, Petrovic S, Qu R. Case-based reasoning in course timetabling: an attribute graph approach. Lecture notes in computer science, vol. 2080, Berlin: Springer; 2001. p. 90–104.
[12] Burke EK, Newall JP. Solving examination timetabling problems through adaption of heuristic orderings. Annals of Operations Research 2004;129(1-4):107–34.
[13] Ross P, Hart E, Corne D. Genetic algorithms and timetabling. Natural computing series, advances in evolutionary computing: theory and applications. 2003. p. 755–77.
[14] Carrasco MP and Pato MV. A multiobjective genetic algorithm for the class/teacher timetabling problem. Lecture notes in computer science, vol. 2079, Berlin: Springer; 2001. p. 3–17.
[15] Fernandes C., Caldeira JP, Melicio F, Rosa AC. High school weekly timetabling by evolutionary algorithms. In: Symposium on applied computing, Proceedings of the 1999 ACM symposium on applied computing. 1999. p. 344–50.
[16] Nedjah N, de Macedo Mourelle L. Evolutionary time scheduling. In: Proceedings of the international conference on information technology, coding and computing (ITCC'04), vol. 2, 2004. p. 357–61.
[17] Di Gaspero L, Schaerf A. Tabu search techniques for examination timetabling. Lecture notes in computer science, vol. 2079, Berlin: Springer; 2001. p. 104–17.
[18] Burke EK, Kendall G, Soubeiga E. A tabu-search hyperheuristic for timetabling and rostering. Journal of Heuristics 2003;9(6):451–70.
[19] Socha K, Knowles J, Sampels M. A MAX–MIN ant system for the university course timetabling problem. Lecture notes in computer science, vol. 2463, Berlin: Springer; 2002. p. 1–13.
[20] Rossi-Doria O, Sampels M, Birattari M, Chiarandini M, Dorigo M, Gambardella LM, et al. A comparison of the performance of different metaheuristics on the timetabling problem. Lecture notes in computer science, vol. 2740, Berlin: Springer; 2003. p. 329–51.
[21] Smith KA, Abramson D, Duke D. Hopfield neural networks for timetabling: formulations, methods, and comparative results. Computers and Industrial Engineering 2003;44(2):283–305.
[22] Cambazard H, Demazeau F, Jussien N, David P. Interactively solving school timetabling problems using extensions of constraint programming. Lecture notes in computer science, vol. 3616, Berlin: Springer; 2005. p. 190–207.
[23] Wren A. Scheduling, Timetabling and rostering—a special relationship. The practice and theory of automated timetabling. Lecture notes in computer science, vol. 1153, Berlin: Springer; 1996, p. 46–76.
[24] Schaerf A. Local search techniques for large high school timetabling problems. IEEE Transactions on Systems, Man and Cybernetics—Part A: Systems and Humans 1999;29(4):368–77.
[25] Ten Eikelder HMM, Willemen RJ. Some complexity aspects of secondary school timetabling problems. Lecture notes in computer science, vol. 2079, Berlin: Springer; 2001. p. 18–27.
[26] Papoutsis K, Valouxis C, Housos E. A column generation approach for the timetabling problem of greek high schools. Journal of the Operational Research Society 2003;54(3):230–8.
[27] Valouxis C, Housos E. Constraint programming approach for school timetabling. Computers and Operations Research 2003;30(10):1555–72.
[28] Caldeira JP, Rosa A. School timetabling using genetic search. In: Proceedings of the second international conference on the practice and theory of automated timetabling (PATAT'97), 1997. p. 115–22.
[29] Fernandes C, Caldeira JP, Melicio F, Rosa AC. Evolutionary algorithm for school timetabling. In: Proceedings of the genetic and evolutionary computation conference (GECCO' 99), vol. 2, 1999. p. 1777–83.
[30] Bufé M, Fischer T, Gubbels H, Häcker C, Hasprich O, Scheibel C, et al. Automated solution of a highly constrained school timetabling problem-preliminary results. Lecture notes in computer science, vol. 2037, Berlin: Springer; 2001. p. 431–40.
[31] Tavares R, Teofilo A, Silva P, Rosa A. Infected genes evolutionary algorithm. In: Symposium on applied computing (SAC'99), 1999. p. 333–8.
[32] Santiago-Mozos R, Salcedo-Sanz S, DePrado-Cumplido M, Bousoño-Calzón C. A two-phase heuristic evolutionary algorithm for personalizing course timetables: a case study in a Spanish University. Computers and Operations Research 2005;32(7):1761–76.
[33] Di Stefano C, Tettamanzi AGB. An evolutionary algorithm for solving the school timetabling problem. In: Boers EJW et al. editors. Lecture notes in computer science, vol. 2037, Berlin, Heidelberg: Springer; 2001. p. 452–62.
[34] Filho GR, Lorena LAN. A constructive evolutionary approach to school timetabling. Lecture notes in computer science, vol. 2037, Berlin: Springer; 2001. p. 130–39.
[35] Wilke P, Gröbner M, Oster N. A hybrid genetic algorithm for school timetabling. Lecture notes in computer science, vol. 2557, Berlin: Springer; 2002. p. 455–64.
[36] Alvin C, Kwan M, Ken C, Chung K, Kammy K, Yip K, Tam V. An automated school timetabling system using hybrid intelligent techniques. Lecture notes in computer science, vol. 2871, Berlin: Springer; 2003. p. 124–34.
[37] Beligiannis GN, Moschopoulos CN, Likothanassis SD. A genetic algorithm approach to school timetabling. Journal of the Operational Research Society, submitted for publication.
[38] Ross P, Corne D, Fang H. Successful lecture timetabling with evolutionary algorithms. In: Proceedings of the workshop on applied genetic and other evolutionary algorithms (ACAI'94), 1994.
[39] Goldberg DE. Genetic algorithms in search, optimization and machine learning. Reading, MA: Addison-Wesley; 1989.

[40] Michalewicz Z. Genetic algorithms + data structures = evolution programs. 3rd ed., New York: Springer; 1996.
[41] Blickle T, Thiele L. A comparison of selection schemes used in genetic algorithms. Evolutionary Computation 1997;4(4):361–94.
[42] Mitchell M. An introduction to genetic algorithms. Cambridge, MA, London, UK: The MIT Press; 1995.
[43] Bäck T. Evolutionary algorithms in theory and practice. Oxford: Oxford University Press; 1996.
[44] Reeves CR, Wright CC. Epistasis in genetic algorithms: an experimental design perspective. In: Proceedings of the sixth international conference on genetic algorithms, 1995. p. 217–24.