A Multiobjective Genetic Algorithm for the Class/Teacher Timetabling Problem

Marco P. Carrasco[1,3] and Margarida V. Pato[2,3]

[1] Escola Superior de Gestão, Hotelaria e Turismo, Algarve University,
Largo Eng. Sárrea Prado n.21, 8500 859 Portimão, Portugal,
pcarras@ualg.pt,
[2] Instituto Superior de Economia e Gestão, Technical University of Lisbon,
Rua do Quelhas n.6, 1200 781 Lisboa, Portugal,
mpato@iseg.utl.pt
[3] Centro de Investigação Operacional, Faculdade de Ciências, University of Lisbon

Abstract. The drawing up of school timetables is a slow, laborious task, performed by people working on the strength of their knowledge of resources and constraints of a specific institution. This paper begins by presenting the timetabling problems that emerge in the context of educational institutions. This is followed by a description of the basic characteristics of the class/teacher timetabling problem. Timetables are considered feasible provided the so-called hard constraints are respected. However, to obtain high-quality timetabling solutions, other conditions should be satisfied in this case – those of soft constraints – which impose satisfaction of a set of desirable conditions for classes and teachers. A multiobjective genetic algorithm was proposed for this timetabling problem, incorporating two distinct objectives. They concern precisely the minimization of the violations of both types of constraints, hard and soft, while respecting the two competing aspects – teachers and classes. A brief description of the characteristics of a genetic multiobjective metaheuristic is presented, followed by the nondominated sorting genetic algorithm, using a standard fitness-sharing scheme improved with an elitist secondary population. This approach represents each timetabling solution with a matrix-type chromosome and is based on special-purpose genetic operators of crossover and mutation developed to act over a secondary population and a fixed-dimension main population of chromosomes. The paper concludes with a discussion of the favorable results obtained through an application of the algorithm to a real instance taken from a university establishment in Portugal.

1 Introduction

In most educational institutions, the timetabling activity assumes considerable importance. First, as a result of this task, a set of attributions determining the entire daily activity of all the human resources involved, from students and teachers to employees, is drawn up. Secondly, alongside, a pattern of the use of the institution's physical resources is defined. Rational management of such

E. Burke and W. Erben (Eds.): PATAT 2000, LNCS 2079, pp. 3–17, 2001.

resources is of major importance at the pedagogical and administrative level. Usually, the timetabling data changes substantially from year to year and consequently the previous timetable cannot be used as a starting solution. So this complex task of building school schedules is performed from scratch, each year, by a team of technicians. This is based on the knowledge and experience acquired of the institution and leads to a set of acceptable schedules, through a trial and error procedure.

Distinct types of timetabling problems emerge with regard to schools, and depend on the elements to be scheduled (exams or regular courses). This paper focuses on a particular regular course problem: that of generating weekly timetables involving the assignment of a set of lessons (assignments of teachers to classes within the respective subjects) to classrooms and time periods – the class/teacher timetabling problem (CTTP).

In the operation research domain the CTTP has led to numerous approaches (de Werra 1985, 1997). In fact, this timetabling problem substantially differs from institution to institution, not only in view of the set of conditions required for the final solution, but also due to the dimension of the respective instances. Moreover, as a rule such situations generate high-complexity optimization problems. A practical solution is difficult to encounter when instances are highly dimensioned, which accounts for the application of heuristics to the field. Within this topic, the evolutionary approach provided by genetic algorithms pointed to interesting ways of tackling timetabling problems.

In this paper, a multiobjective genetic heuristic for the CTTP is explored. Section 2 presents the problem under study by describing its main characteristics, whereas Section 3 summarizes the basic foundations of the genetic algorithms as well as referring to a special multiobjective genetic algorithm – the nondominated sorting genetic algorithm (NSGA). Adaptation of the NSGA to the CTTP is discussed in Section 4, and Section 5 shows the computational results obtained from application of the proposed model to a real instance. Finally, Section 6 closes the paper with some conclusions and significant considerations.

2 The Class/Teacher Timetabling Problem

Generally, in teaching institutions two main types of timetabling problems arise: one for exams (the examination timetabling problem) and the other for regular courses (consisting of the course timetabling problem and the CTTP).

The examination timetabling problem is similar in most schools and consists of scheduling the exams for a set of courses, over a limited time period, while avoiding the overlapping of exams for each student, as well as seeking the largest spread over the examination period. A survey of this problem may be found in Carter and Laporte (1996) or in Burke et al. (1996). As to regular courses, the Course Timetabling Problem, typical of universities with flexible curricula, sets out to schedule all lectures included within the set of the institution's courses, while minimizing the overlaps of lectures for courses with common students (see, for instance, Downsland (1990); Kiaer and Yellen (1992)).

The other problem, which is precisely the subject of this study, the CTTP, is usually found in schools with less modular curricula. Such is the case in most high schools and some universities, where the majority of the predefined lessons ascribed to each course are compulsory for the respective classes. Unlike the course timetabling problem, where the CTTP is concerned, the overlapping of lessons is not allowed. Here, the goal consists of scheduling the set of lessons while satisfying a range of constraints specific to the institution. As a result, the CTTP itself emerges in slightly diverse forms and dimensions according to the features of the school (curricula, number of teachers and students, number of rooms and characteristics, geographical layout of the buildings, history of scheduling procedures, etc.) and system (high school or university). The specific problem also depends on the rules imposed by the national educational policy. This issue has always been formulated through combinatorial optimization problems that are very difficult in terms of complexity. In fact, even the simplified versions of the CTTP proved to be NP-hard (Even et al. 1976).

The above considerations and the high-dimension instances normally found in real timetabling problems may explain the continuous interest in studying the CTTP. In fact, several resolution techniques have been applied, from graph-theory-based algorithms (de Werra 1996), to binary programming (Tripathy 1984; Birbas et al. 1997) and constraint-based approaches (Yoshikawa et al. 1994). Moreover, meta-heuristics based on simulated annealing (Abramson 1982), neural networks (Gislén et al. 1992), genetic algorithms (Colorni et al. 1991) and tabu search (Costa 1994) have all been applied with success to specific cases, under study by the respective authors.

In general terms, the CTTP described in this paper resembles all other CTTPs and can be defined as the problem of scheduling a set of lessons (here, a lesson is a prior assignment of one or more rigid classes of students to a teacher and a subject) over a set of time periods and suitable rooms, while satisfying a wide range of constraints. These can be divided into two levels of importance: the hard constraints and the soft constraints.

1. The hard constraints represent compulsory restrictions associated with the feasibility of a solution for the CTTP:
   (a) Each teacher and class is assigned to no more than one lesson and one room at a time period.
   (b) Each class curriculum is respected, i.e. all lessons are scheduled.
   (c) The daily bound on the number of lessons for each subject is not exceeded due to pedagogical reasons.
   (d) Rooms are suitable for the lessons assigned. Each lesson may require a specific type of room with enough seats and/or special resources.
   (e) The teaching shift for each class is respected.
2. The soft constraints represent a second level of non-compulsory restrictions, which refers to aspects that both teachers and classes would like to see satisfied. These conditions are desirable and closely related to the final quality of the solution: a high-quality timetabling solution should satisfy the maximum number of these conditions:

(a) Each class and teacher should have a lunch break lasting at least one hour.
(b) The occurrence of gaps between teaching periods should be minimal, for classes and teachers.
(c) Class and teacher preferences should be satisfied, whenever possible.
(d) The number of teacher and class shifts between teaching locations should be minimized.
(e) The number of days occupied with lessons should be minimized.

These are the basic features considered in our methodology to solve the CTTP. Other particular conditions may be incorporated in the algorithm described in detail in Section 4.

3 Multiobjective Genetic Algorithms

During the last decade, evolutionary techniques such as genetic algorithms (GAs) have been extensively used to solve complex optimization problems. The reference model introduced by John Holland (1975) triggered a wide interest in the application of these types of heuristics, which mimic the natural evolutionary characteristics present in the biological species, with the purpose of solving optimization problems efficiently. Several evolutionary approaches rely on modifications made to Holland's initial model, while preserving the essential evolutionary characteristics. Next, there follows a description of the basic aspects of a standard GA for multiobjective optimization.

The conception of a GA necessarily begins with the choice of the chromosome encoding for each individual, which is the structure that represents each solution of the problem to be solved. Traditionally the encoding is the binary one, theoretically supported by the schema theorem formulated by Holland. In practice, albeit applicable to any problem, in some issues the binary encoding is not the most suitable one. It leads to significant computing time expense due to additional readjustment procedures. For this reason, some authors propose more natural representations, i.e. non-binary alphabets (Goldberg 1989). These problem-oriented encodings may improve certain aspects of the GA, such as the search ability for good solutions. They may also enable one to conceive genetic operators which are specifically suited to the problem and require no hard time-consuming readjustments.

Having defined the appropriate representation for the solutions of the optimization problem, an initial population of individuals or respective chromosomes is generated. This is normally achieved by resorting to random procedures. Next, a fitness function must be created to evaluate each chromosome. This function assigns a fitness value to each chromosome according to the quality of the respective solution. In general it reflects the level of optimization of the problem objective for the corresponding solution.

In conventional GAs, evaluation of the individuals reflects only the optimization of a single objective, but the nature of some real-world optimization problems may benefit from an approach guided by more objectives. Habitually,

in the interest of the solution, the multiple objectives are in some way combined through an aggregating function that is a linear combination of the objective functions. However, when the different objectives are not comparable and mixable in terms of penalization, or the definition of the penalizing coefficients is difficult because some objectives tend to dominate the others, this approach seems inappropriate.

To tackle multiobjective optimization problems by using GAs, several other methodologies have been applied, as mentioned in detail by the following surveys: Fonseca and Fleming (1995), Veldhuizen and Lamont (2000). The most widely utilized are the so-called Pareto-based techniques employing the concept of Pareto optimality to evaluate each chromosome in relation to the multiple objectives. The algorithm used in the current paper was derived from the NSGA proposed by Srinivas and Deb (1994), which applies the original Pareto-based ranking procedure implemented by Goldberg (1989) combined with a fitness sharing scheme. The main distinction consists in the implementation of a secondary population procedure to improve the solution quality.

The population is then subjected to the genetic operators of selection, crossover and mutation. During the selection phase, pairs of individuals are chosen according to their fitness. These pairs are submitted to a crossover operator, which, through a combination scheme, generates a new pair that may substitute its progenitors in the population. The mutation operator then introduces random, sporadic modifications, that is mutations, in the genetic content of some randomly chosen chromosomes.

All chromosomes of the population for the next generation are evaluated once more and the procedure is repeated up to a maximum number of generations. Here the intention is to gradually improve the quality of the population. Together, these operators support the search mechanism of a GA by seeking the good solutions suggested by the fitness function.

The structure of a generic NSGA is summarized in Figure 1. Note that the final result of a NSGA may not be one single best solution but a set of Pareto front solutions.

4 The NSGA for the CTTP

The CTTP is, as mentioned above, a complex problem which, within the current approach, assumes the configuration of a multiobjective optimization problem with two main objectives. Nevertheless, it should be noted that this approach is suitable for incorporating more than two objectives such as administration, space workload or security objectives. In fact, following a detailed inspection of the nature of the so-called soft constraints, one may see that they compete with each other for teachers and classes. For instance, a tentative imposition of only the soft constraints related to the teachers would certainly produce timetables with high violations for classes. To prevent this and simultaneously enforce satisfaction of the hard constraints, a multiobjective GA is devised according to the lines of Figure 1, with two distinct objectives conceived to express CTTP con-

BEGIN
Step 1 ▷ **Initialization**
Setup of an initial population of N chromosomes
Step 2 ▷ **Evaluation**
Calculation of objective values for each chromosome
Updating of Pareto front and rankings
Evaluation of all chromosome's fitness
Undertaking of fitness sharing
Step 3 ▷ **Selection, Crossover and Mutation**
Application of genetic operators
Replacement of chromosomes by offspring
⇑ **Loop to Step 2** ▷ Until the maximum number of generations is reached
END (Output ← Pareto front chromosomes)

Fig. 1. Outline of a generic NSGA

straint violations associated mainly with classes and teachers, respectively. The algorithm will attempt to find a set of feasible and high-quality timetabling solutions. This is performed through a constraint violation penalizing scheme that distinctively penalizes the hard and soft constraints. The penalizations were defined by using a trial and error procedure, which indicated that the algorithm is quite sensitive to the selection of these parameters. Moreover, the method deals with a main population as well as an elitist secondary population.

4.1 Chromosome Encoding

In practice, timetabling plans for the resources involved in a CTTP, such as classes, teachers or rooms, are usually represented in matrix form, where at least one of the axes always represents a time scale. This idea led to the option of a natural representation of the CTTP chromosome, encoded through a non-binary alphabet within a matrix. Although one could also use a binary encoding for the CTTP, this choice would require an additional effort to verify and impose the constraints.

Let the number of available rooms and time periods be R and T, respectively, and each integer element of the $R \times T$ matrix a label that represents a lesson (class(es)/teacher/subject) assigned to the respective room (row) and time period (column). Here, lessons are the basic elements to be scheduled and can be of different durations as well as having distinct requirements.

As an illustration, consider the curricula of two classes consisting of 12 lessons, presented in Table 1, where each row corresponds to a lesson and specifies the class(es) and teacher involved, the respective subject, duration and a list of required rooms. Figure 2, exemplifies a chromosome matrix with 3 rows (rooms) and 20 columns (time periods) relative to these curricula.

This specific chromosome encoding allows one to aggregate the corresponding timetables for classes and teachers in a single matrix and all the lessons previously defined in the curricula of classes are scheduled within the matrix

Table 1. Example of curricula

Lesson	Classes	Teacher	Duration	Subject	Room required
M1	c_1 and c_2	t_1	1 period	Maths	r_2
M2	c_1	t_1	2 periods	Maths	r_1 or r_2
M3	c_2	t_1	2 periods	Maths	r_1 or r_2
E1	c_1	t_2	2 periods	Economics	r_3
E2	c_2	t_2	1 period	Economics	r_3
E3	c_1 and c_2	t_3	2 periods	Economics	r_2
H1	c_1	t_2	1 period	History	r_3
H2	c_2	t_2	1 period	History	r_3
H3	c_2	t_2	1 period	History	r_3
S1	c_1	t_1	1 period	Statistics	r_1 or r_3
S2	c_2	t_1	1 period	Statistics	r_1 or r_3
S3	c_1 and c_2	t_1	1 period	Statistics	r_2

chromosome. Thus, each chromosome represents a solution for the CTTP, i.e.

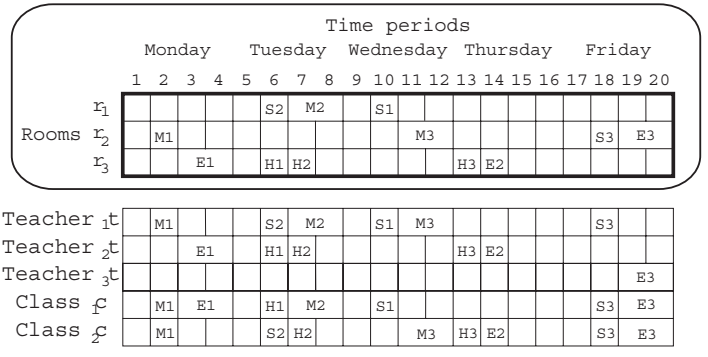

Fig. 2. Example of chromosome encoding ($R \times S$ matrix) and corresponding timetables

a set of timetables, that automatically satisfies condition 1(b). However, it may include violations for the other hard constraints, thus representing a non-feasible solution.

The conception of this multiobjective GA is to search for high-quality solutions – solutions violating at the minimum the soft constraints – while penalizing severely violations of hard constraints, i.e. non-feasible solutions. There are two main reasons for this option. First, the approach is very flexible and permits swift implementation of further constraints that may appear in other real CTTPs. The second reason concerns the computational effort to maintain

a population of feasible solutions, requiring the expenditure of significant computing time when repairing the non-feasible solutions produced by the genetic operators. In this way, the task of forcing feasibility is performed gradually via a constraint penalization scheme.

4.2 Fitness Function

As observed above, in the current development for the CTTP, two competitive objectives emerge from the soft and hard constraints described in Section 2:

- minimization of the penalized sum of violations mainly occurring in the constraints referring to the teachers, teacher-oriented objective;
- minimization of the penalizated sum of violations occurring in the constraints mainly related to classes, class-oriented objective.

Next, there follow some details for the calculation of these two objective function values for each solution, i.e. a set of timetables encoded through a matrix chromosome. Assuming one has a chromosome, the satisfaction of those aspects included in the soft constraints (2(a)–(e)) and hard constraints (1(a), 1(c), 1(d) and 1(e)) is checked. The amount of violations is counted (both for teachers and classes) and, by using a constraint violation penalization scheme, the respective values of the two objectives considered are produced. However, the penalization procedure deals with hard and soft constraints differently. As for soft constraints, which are non-obligatory for timetabling feasibility, the penalizations assigned are small compared with those of hard constraints, thus reflecting solution unfeasibility. In addition, the violations of hard constraints are not linked with teachers or classes alone, but with the solution globally. Therefore, the penalizations of the hard constraints are added to both objective values simultaneously.

At first, each solution of the population is classified in terms of dominance by other solutions. As usual, a nondominated solution does not allow the existence of any other better for all the objectives. In such a case, any improvement on one of the objectives must worsen at least one of the remaining ones. Then, through a ranking process, due to Goldberg 1989, the solution fitness is determined according to the relative solution dominance degree. Basically the Pareto front, formed by the nondominated solutions, receives rank 1 and, from the remaining population, the next nondominated solutions receive rank 2 and so on, until the entire population is ranked. Finally, the fitness of each chromosome is obtained by inverting the chromosome rank level.

The fitness assignment method described above may lead to the loss of diversity within the population due to stochastic errors in the sampling selection procedure. In fact, when faced with multiple equivalent Pareto front solutions, the populations tend to converge to a single solution as a result of the genetic drift. To reduce this effect and keep the Pareto front uniformly sampled, the NSGA includes a procedure of fitness sharing. This is produced by dividing the Goldberg fitness $f(x_i)$ by a value that reflects the number of similar chromo-

somes around x_i, within the same rank level, as follows:

$$f(x_i)_{shared} = \frac{f(x_i)}{\sum_{j=1}^{N} s[d(x_i,x_j)]} \quad . \tag{1}$$

Here, N is the total size of main and secondary populations. The function $s\left[d\left(x_i, x_j\right)\right]$ states, for the distance between two given chromosomes $d\left(x_i, x_j\right)$, the intensity of the sharing process:

$$s\left[d\left(x_i, x_j\right)\right] = \begin{cases} 1 - \left(\frac{d(x_i,x_j)}{\delta_{share}}\right)^{\alpha} & \text{if } d\left(x_i, x_j\right) < \delta_{share} \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where δ_{share} defines the frontier of the neighborhood and α acts as a strength-sharing parameter. The measure of distance $d\left(x_i, x_j\right)$ is evaluated in the genotypic space by analyzing the similarities of the pair of solutions depending on the use of the same time period, for each lesson:

$$d(x_i, x_j) = \frac{\sum_{l=1}^{L} t(l, x_i, x_j)}{L} \quad , \tag{3}$$

where

$$t(l, x_i, x_j) = \begin{cases} 1 \text{ if lesson } l \text{ occupies the same time period in solution } x_i \text{ and } x_j, \\ 0 \text{ otherwise,} \end{cases} \tag{4}$$

and L is the total number of lessons to be scheduled.

Other alternative measures of distance can be applied, as proposed by Burke et al. (1998), with distinct implications on the global diversity of the population. Due to the large number of lessons often involved in the CTTP and the resulting computational effort required to compute the similarity within the population, we opted for the above absolute position distance measure.

The fitness sharing procedure is applied to each chromosome rank by rank. Through the assignment of dummy fitness values, the minimum fitness value of the current rank is always higher than the maximum fitness value of the next rank.

4.3 Main and Secondary Populations

The main population of 200 chromosomes is initially built by using a constructive random heuristic. The heuristic procedure relative to each chromosome starts by scheduling those lessons that are more difficult to allocate, according to a previously defined sorted list, while trying to satisfy the maximum of constraints. This degree of difficulty is measured in terms of lesson duration, preferences of teacher and classes involved and room requirement. The longer and more resource-demanding lessons are usually more difficult to schedule. This ends up with a chromosome that is a set of timetables satisfying at least the hard constraint 1(b).

Due to the stochastic nature of a GA, some high-quality solutions can be definitely lost during the algorithm execution as a result of sampling. Hence, the use of an elitist secondary population composed of some of the Pareto front solutions found so far, may increase the performance of the multiobjective GA (Zitzler 1999). The developed NSGA for the CTTP includes a secondary population, as illustrated in Figure 3. At the beginning of generation k, suppose

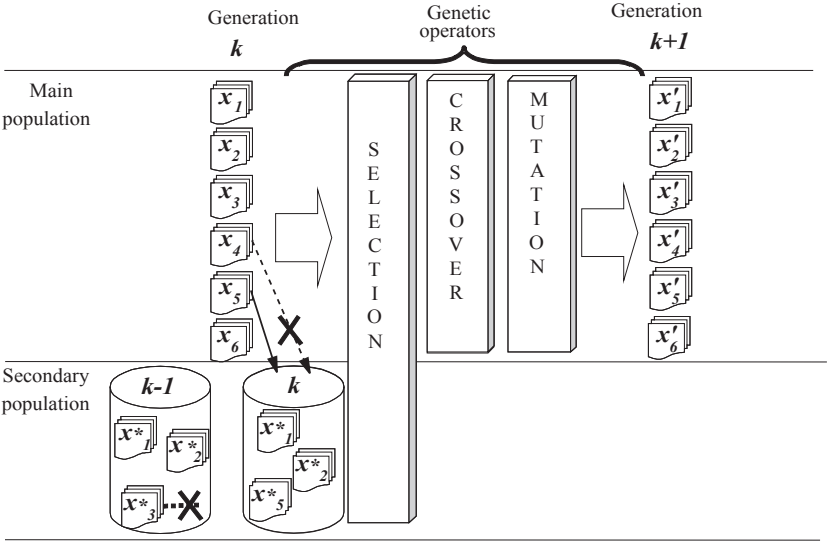

Fig. 3. Example of main and secondary populations updating scheme

that the main population is formed by six chromosomes (x_1 to x_6), and the secondary population includes three chromosomes (x_1^* to x_3^*). After the ranking process within the main population, performed for the fitness calculation, the non-dominated solutions (assume they are x_4 and x_5) are candidates to enter the secondary population. However, they do not always enter, but only if they are sufficiently distinct from the solutions that are stored therein. Suppose that x_4 is very similar to x_2^*, according to the measure $d(x_4, x_2^*)$, then it must be rejected. Next, each secondary population chromosome retained from generation $k-1$ (x_1^* to x_3^*) and candidate chromosome (x_5) is checked for non-dominance, keeping only the non-dominated ones in the secondary population for generation k. In this example, assume that x_5 dominates x_3^*, thus causing the removal of x_3^*.

4.4 Selection, Crossover and Mutation Operators

Operating over the above main and secondary populations of timetabling chromosomes and using a standard roulette wheel procedure, the selection operator chooses 100 pairs of chromosomes to be combined by the crossover.

As for the crossover, a customized genetic operator is created to effectively combine the contents of a pair of parent chromosomes. Figure 4 will be used to describe this procedure. Having chosen two chromosomes from the above

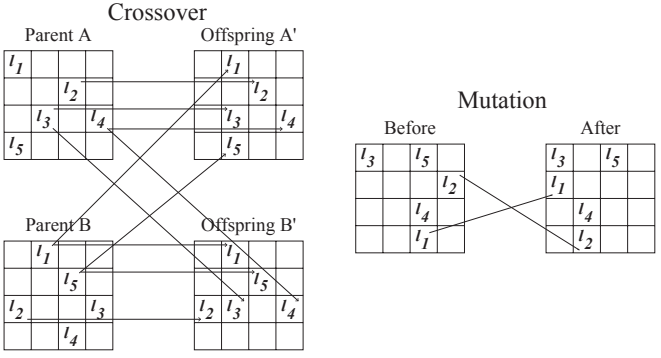

Fig. 4. Example of crossover and mutation operator procedures

populations (parent A and parent B), with the selection operator, two A′ and B′ offspring are created. The procedure begins by randomly defining a number c of lessons to be received through inheritance by each offspring, within 80 to 95% of the total number of lessons. The offspring A′ is obtained from parent A selecting c lessons that belong to the teachers with the best timetables (according to the partial value of constraint violation penalizations for each teacher). Suppose that these lessons are represented in Figure 4 by l_2, l_3 and l_4. The remaining lessons are taken from parent B (l_1 and l_5) and are eventually subjected to a repair procedure to avoid overlapping. In this example the location of l_5 was already occupied by l_2. Consequently, a new location for l_5 must be found. Offspring A′ is therefore teacher-guided, because it includes those lesson assignments that benefit the teachers. Offspring B′ is obtained through a similar process but for classes. Therefore B′ is, in this way, class-guided. Both offspring A′ and B′ are inserted in the transitory population during 100 crossover operations.

This specific crossover operator, which continuously transfers to the offspring a fraction of the best timetables produced so far, both for teachers and classes, strives to gradually improve the two distinct objectives within the population.

The mutation operator illustrated in Figure 4, can be described as follows. For each chromosome of the transitory population created by selection and crossover, the mutation procedure randomly takes m lessons. This amount of lessons may

vary up to 1% of the total number of lessons L. In the example, where m equals two, consider that these lessons are represented by l_1 and l_2. Such lessons are removed from the actual locations and then, from the set of the remaining locations, the procedure finds new locations that cause the minimal increase in total penalization.

At this point, the current generation ends and the transitory population is moved to the main population for the next generation. Note that the GA parameters were set by trial and error and remained appropriate to all the CTTP instances tackled.

5 Computational Results

The above-described algorithm was tested on real timetabling situations, namely the first semester instance of the Escola Superior de Gestão, Hotelaria e Turismo of the Algarve University, for the last academic year. Experiments with other instances also tested from this school showed similar behavior.

This instance is characterized in the following way: 716 lessons to schedule, 29 available rooms on two distinct campuses, 108 teachers, 40 classes and a week of 50 time periods.

The proposed algorithm was applied from the initial main population of 200 timetabling solutions until the maximum number of 1000 generations was reached. It was tested in four different versions. The idea behind this was to test the effect of the fitness sharing procedure and the secondary population on the quality of the final solutions: versions 3, 4 with a secondary population and versions 2, 4 with fitness sharing, corresponding to NSGAs. Version 1 was used for purposes of comparison.

The population of Pareto front timetables obtained at the end of a single run of each GA version is shown in Figure 5. Also represented is the hand-made solution, equally expressed in terms of teacher and class-oriented penalizations. Note that the lowest Pareto front, representing the best solutions for teachers, came from version 3. As shown in this figure, the version 4 algorithm provided the best solutions for classes.

In addition, Tables 2 and 3 present some details on the computational results obtained from each algorithm version.

As for the number and feasibility of Pareto front solutions, version 3 generated the largest set containing 81 timetabling solutions, 76 of which satisfied all hard constraints (columns 5 and 6 of Table 2). When analyzing column 4 of the same table, in reporting the mean similarity degree that expresses how repeatedly each lesson is assigned to the same time period within the Pareto front solutions, one can see that the lowest value was obtained for version 4, with an average use of 34% of the same time periods. Such behavior is due to the fitness sharing that produces a small level of similarity in versions 2 and 4. Also, both versions show a wide and spread Pareto front set, while versions with the secondary population scheme, versions 3 and 4, obtained a more numerous Pareto

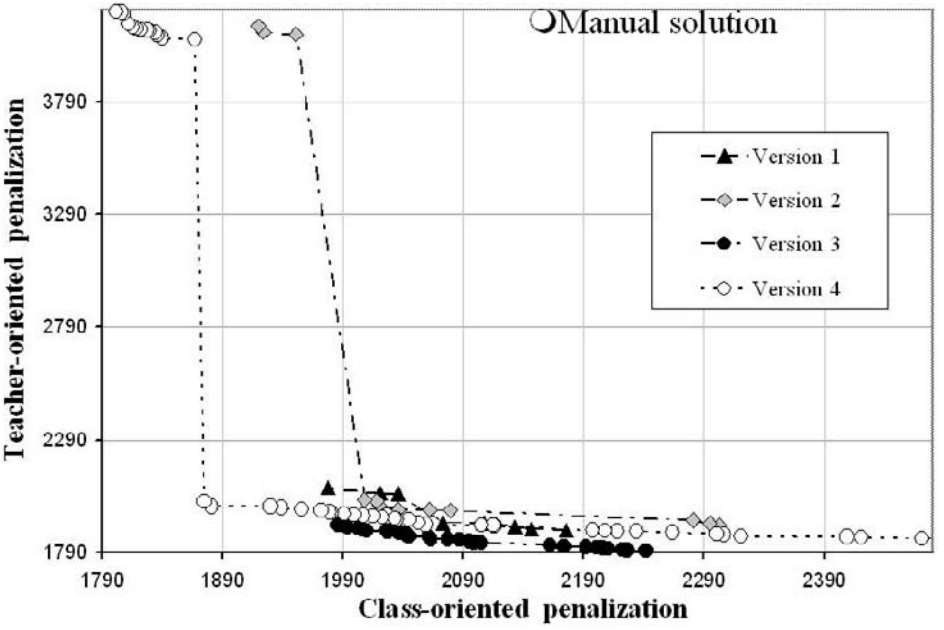

Fig. 5. Pareto front results for the algorithm versions

front. In fact, algorithm versions 3 and 4 achieved the largest sets of feasible solutions (76 and 55 solutions compared with the 12 and 14 solutions obtained in versions 1 and 2, respectively), allowing the decision-maker to choose the more appropriate ones. A detailed analysis of the set of feasible solutions, referred to in column 6 of Table 2, expressed in terms of soft constraints violations, is presented for each algorithm version, in Table 3. Columns 2 to 6 show the average number of soft constraints not satisfied (violated) per feasible solution, while columns 7 and 8 present the average penalizations relative to the set of feasible solutions obtained for classes and teachers. The results clearly point to the fact that versions 3 and 4 with a secondary population are superior to the standard NSGA and to the simplified GA (versions 2 and 1 respectively), both in terms of soft constraint violations and solution penalizations. In all cases the execution time to complete the 1000 generations was similar, about 4 h on a Pentium III 500Mhz PC with 64 MB of memory. It should be added that the execution time is compatible with the situation tackled, whereas the construction of this manual solution involved one week's work by a three person team.

Finally, comparison of the solutions obtained by the algorithm versions with the manual solution (displayed graphically in Figure 5) demonstrates that almost all solutions are superior in the two objectives to the manual solution.

Table 2. Algorithm versions: similarity, feasibility and execution time

Version			Pareto front solutions			Execution
Number	Secondary population	Fitness sharing	Mean similarity degree	No.	No. Feasible	time (min)
(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	No	No	66%	13	12	213
2	No	Yes	39%	20	14	225
3	Yes	No	67%	81	76	249
4	Yes	Yes	34%	78	55	261

Table 3. Pareto front feasible solution: mean soft constraint violation

Algorithm	Soft constraints violations					Class	Teacher
version	2(a)	2(b)	2(c)	2(d)	2(e)	penalization	penalization
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
1	0	297	6.2	13.8	405.6	2056.25	2339.25
2	0	277	3.0	20.3	429.1	2074.75	2505.17
3	0	212	1.5	6.0	392.6	1971.88	2103.51
4	0	210	8.1	8.1	399.2	2004.35	2370.21

6 Final Remarks

Overall, the results obtained from the proposed NSGA with the elitist secondary population on the CTTP are promising when applied to a real instance, and lead to the following conclusions. First, the algorithm was able to successfully optimize the two distinct objectives considered. It thus achieved a set of quality timetabling solutions provided, at the end, by the Pareto front solution set. Secondly, this algorithmic approach produced much better timetables than the actual manual timetables, with a significantly reduced effort. On the basis of this algorithm, a software application (named GAHOR) is under development to edit, manage and automatically solve real problems.

In future the authors intend to apply this multiobjective GA to solve much larger CTTP instances, which are already being tackled. This will be attained through the refinement of the algorithm's performance by using improved genetic operators, while taking into account the parallel capability of GAs in order to decrease computational time.

References

Abramson, D.: Constructing School Timetables Using Simulated Annealing: Sequencial and Parallel Algorithms. Manage. Sci. **37** (1982) 83–113

Birbas, T., Daskalaki, S., Housos, E.: Timetabling for Greek High Schools. J. Oper. Res. Soc. **48** (1997) 1191–1200

Burke, E.K., Elliman, D.G., Weare, R.: Examination Timetabling in British Universities – A Survey. In: Burke, E.K, Ross, P. (eds.): Practice and Theory of Automated Timetabling. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg New York, **1153** (1996) 76–90

Burke, E.K., Newall, J., Weare, R.: Initialization Strategies and Diversity in Evolutionary Timetabling. Evol. Comput. **6** (1998) 81–103

Carter, M.W., Laporte, G.: Recent Developments in Practical Examination Timetabling. In: Burke E.K, Ross P. (eds.): Practice and Theory of Automated Timetabling. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg New York, **1153** (1996) 3–21

Colorni, A., Dorigo, M., Maniezzo, V.: Genetic Algorithms and Highly Constrained Problems: The Timetabling Case. Proc. 1st Int. Conf. on Parallel Problem Solving from Nature. Lecture Notes in Computer Science, **496** (1991) 55–59

Costa, D.: A Tabu Search Algorithm for Computing an Operational Timetable. Eur. J. Oper. Res. **76** (1994) 98–110

de Werra, D.: An Introduction to Timetabling. Eur. J. Oper. Res. **19** (1985) 151–162

de Werra, D.: Some Combinatorial Models for Course Scheduling. In: Burke, E.K., Ross, P. (eds.): Practice and Theory of Automated Timetabling. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg New York, **1153** (1996) 296–308

de Werra, D.: The Combinatorics of Timetabling. Eur. J. Oper. Res. **96** (1997) 504–513

Dowsland, K.A.: A Timetabling Problem in which Clashes are Inevitable. J. Oper. Res. Soc. **41** (1990) 907–918

Even, S., Itai, A., Shamir, A.: On the Complexity of Timetable and Multicommodity Flow Problems. SIAM J. Comput. **5** (1976) 691–703

Fonseca, C.M., Fleming, P.J.: An Overview of Evolutionary Algorithms in Multiobjective Optimization. Evol. Comput. **3** (1995) 1–16

Gislén, L., Peterson, C., Söderberg, B.: Complex Scheduling with Potts Neural Networks. Neural Comput. **4** (1992) 806–831

Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA (1989)

Holland, J.: Adaptation in Natural and Artificial Systems. Proc. 1st Int. Conf. on Parallel Problem Solving from Nature. University of Michigan Press, Ann Arbor, MI (1975) (Reprinted by MIT Press)

Kiaer, L., Yellen, J.: Weighted Graphs and University Course Timetabling. Comput. Oper. Res. **19** (1992) 59–67

Srinivas, N., Deb, K.: Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evol. Comput. **2** (1994) 221–248

Tripathy, A.: School Timetabling – a Case in Large Binary Integer Linear Programming. Manage. Sci. **30** (1984) 1473–1498

Veldhuizen, D.A., Lamont, G.B.: Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. Evol. Comput. **8** (2000) 125–147

Yoshikawa, M., Kaneko, K., Nomura, Y., Watanabe, M.: A Constraint-Based Approach to High-School Timetabling Problems: A Case Study. Proc. 12th Conf. on Artif. Intell. (1994) 1111–1116

Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Ph.D. Thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (1999)