Infected Genes Evolutionary Algorithm for School Timetabling

C. FERNANDES, J.P. CALDEIRA, F. MELÍCIO, A. ROSA

LaSEEB-ISR-IST, EST-IPS, I.S.E.L.
Techinical University of Lisbon
Av. Rovisco Pais, 1, TN 6.21, 1049-100 Lisboa
PORTUGAL

Abstract: - In this paper we describe a method for generating high school timetables using an Evolutionary Algorithm (E.A.) with a new set of genetic operators inspired on the Infected Genes Evolutionary Algorithm [14]. We show that widening region of infection considered in [7] leads to a very significant increase in the performance of the E.A.. Comparative tests between the E.A. with and without the improved Bad-Gene operators were made using the data of a large Portuguese high school. These clearly show that the improved the Bad-Gene operators result in a significant improvement in performance.

Key-Words: - Evolutionary Algorithms, Timetabling, Infected Genes.

1 Introduction

In this section we briefly describe the characteristics of the school-timetabling problem.

1.1 The School Timetabling Problem

Given c classes, t teachers and r classrooms it is required to build a set of c+t+r timetables satisfying the needs of the school and respecting the restrictions in the assignment of the lessons.
The restrictions or constraints are usually divided in two types - hard and soft [3],[6] ,[12]. The violations of a hard constraint always lead to invalid timetables. The hard constraints considered in this algorithm are:

- Lessons of a class, teacher or classroom cannot be assigned to the same time slot. If this happens we say the lessons are *superimposed*.
- Lessons of a class of the same subject and type cannot be assigned to time slots on the same day.
- A lesson of a class, teacher or classroom can not be assigned to a time slot where an unavailability exists. We define unavailabilities through *preference maps*, which will be discussed later.
- A classroom capacity may not be exceeded.
- The necessary breaks must be respected. Example: lunch break.

The soft constraints are preferences. Their violations affect the quality of the results but do not imply invalid timetables. The soft constraints considered are:

- The gaps between lessons of students and teachers should be minimized.
- Classes and teachers have a limit to the number of consecutive time slots with lessons.
- Classes should either have lessons in the morning shift or afternoon shift.
- The preferences of teachers, classes and classrooms should be respected (see *preference maps*).

1.2 The Data Structures

It is difficult to conceive a data-structure that makes the representation of the many different assignment configurations possible. Some schools have special needs in the assignment of the lessons and a good data structure must be as flexible as possible to enable codification of any situation that may occur. The data-structure used in this E.A. is composed of the following characteristics:

- Lessons of the same subject, duration and type (Laboratory, Practical, Lecture), taught to the same class are grouped together in sets we call *lesson-sets*.
- A Class is divided into subclasses when there are sub-groups of students that attend different lessons.
- Enables the specification of the classroom to which a lesson must be assigned or a set of classrooms from which one is chosen by the algorithm.
- Permits the codification of cases in which more than one class attends a lesson or more than one teacher lectures a class simultaneously.
- Enables the teachers preferences to be taken into account

The most important files of the data structure are the lessons database and the preference maps.

1.2.1 Lesson Database

The lesson database consists of the list of all the information required by each lesson-set. An example of a lesson set codification can be seen in table 1.

Cl	Sub	SC	T	Dur	NL	T	Cl
Cl2	A	2	Lab	180	1	10	Cr98

Cl – Class(es) Sub – subclass(es) SC – Subject Code T – Type	Dur - Duration (in minutes) NL - Nº of lessons T - Teacher(s) Cl - classroom(s) or set of classrooms(s)

Table 1.- Example of codification of a lesson-set in the lesson database

1.2.2 Preference Maps

For every class, teacher, classroom and lesson-set there are maps where, for each time slot, a preference for the assignment of lessons can be defined. The levels of preference decrease from 1 to 4, preference 4 meaning that the time slot is forbidden. For preference 2 and 3 assignment is allowed but increasingly penalised.

	Mon.	Tues.	Wed	Thurs.	Fri.
8.00	1	1	1	1	4
9.00	1	4	1	1	4
10.00	1	4	1	1	4
11.00	1	1	1	1	4
12.00	1	1	1	1	4
13.00	3	3	3	3	4
14.00	3	3	3	3	4
15.00	3	3	3	3	4
16.00	3	3	3	3	4
17.00	3	3	3	3	4

Figure 1 - Preference map of a teacher who prefers lessons in the morning and cannot teach on Fridays.

2 The Algorithm

Before we describe the algorithm some definitions are needed. We define ***time slot duration*** as the greatest common factor of all the different lesson durations i.e. if we only have sixty-minute and ninety-minute lessons then the time slot duration will be thirty minutes. The ***dimension*** (number of time slots) of each timetable is defined as:
dimension=
number_of_days*number_of_time_slots_per_day

2.1 Chromosome Representation

Every chromosome in this algorithm represents a solution to the timetabling problem. Each chromosome therefore contains only the information that varies between different solutions, which is the sequence of [classroom, starting time slot] of each lesson as shown in figure 2. The remaining information required to build the timetables, the constant data, is stored in an array.

Room	Time slot 1	Room	Time slot 2	Room	Time slot 3	Room	Time slot 1	....

Figure 2.- Representation of two lesson-sets - the first with three lessons and the second with only one.

2.2 Initialization

To create the first population of chromosomes the algorithm assigns a classroom and time slot to each lesson. First, if a classroom is not specified in the database one is chosen among those available in the corresponding classroom set. Then, the lesson starting time slot is randomly chosen within those that do not result in a violation of the forbidden timeslots of the corresponding preference maps.

2.2.1 Lessons Assignment Priority

The order in which a lesson is assigned will greatly influence the performance of the algorithm because the smaller the lesson duration, the easier it is to find a place to put them. The lessons are therefore ordered [8] by the following criteria:

[C1]. Duration of each lesson. Longer lessons are assigned first.

[C2]. The ratio between the duration and possible starting time slots. Those with a larger ratio are assigned first.

[C3]. The lessons with predefined classrooms are assigned first. When a block of classrooms is available it is easier to find an empty one.

[C4]. Teachers with a higher priority (supplied by the user) have their lessons assigned first.

2.3 Repair Function

Due to the fact that the creation of new chromosomes with the initialisation or genetic operators is completely random, superimposed lessons are frequent and if a repair function [3] is not used the evolution of the E.A. will be much slower. The repair function implemented in this algorithm is used during the cost evaluation. When superimposed lessons or two lessons with the same code on the same day are found, the function determines all the valid unoccupied time slots to which the lesson that causes the violation can be assigned. The new starting time slot is randomly chosen from those found. If no free time slots are found then the gene value remains unaltered.

2.4 Chromosome Evaluation

The evaluation of each chromosome of the population is done by the following cost function [3], [10]:

$$Cost(chromossome) = C_1 w_1 + C_2 w_2 + ... + C_n w_n$$

where C is the number of violations of a certain constraint and w its corresponding weight.
The fitness function used in this E.A. is the following [2]:

$$F(cost) = k^{\frac{cost\ -\ best\_cost}{average\_cost\ -\ best\_cost}},\ k \in]0,1[$$

Comparative tests showed us that a value of approximately 0.5 for k maximises the performance of the E.A.

2.5 Selection Method and Elitism

The selection of the chromosomes involved in genetic operators is done by *Roulette-wheel selection* [8]. In this method the probability that a chromosome is selected is proportional to its fitness. Elitism was used and tests made with real data showed that the ideal number of chromosomes to pass from one generation to the next is approximately 10% of the population.

2.6 Genetic Operators

In this algorithm we used some typical genetic operators (one point crossover, multipoint crossover, uniform crossover and single gene mutation) and introduced two new operators described below. The probability of each operator is changed according to its relative success rate (ratio between the number of times the operator is used and the number of best chromosomes it generates) of each one.

2.6.1 Bad Gene Operators

In this algorithm bad genes are those that belong to a class, teacher or classroom in which a violation of a hard constraint occurs that cannot be repaired by the repair function. Genes are flagged as *bad* in order to focus the actions of the bad-gene operators.

2.6.1.1 Bad Gene Crossover

This operator chooses two parents randomly and exchanges the bad genes of the better of the two

with the related genes of the other. Classes, teachers and classrooms in which a violation of a hard constraint occurs, that cannot be repaired by the repair function are therefore swapped. In figure 3 we can see the graphic representation of the bad genes crossover.

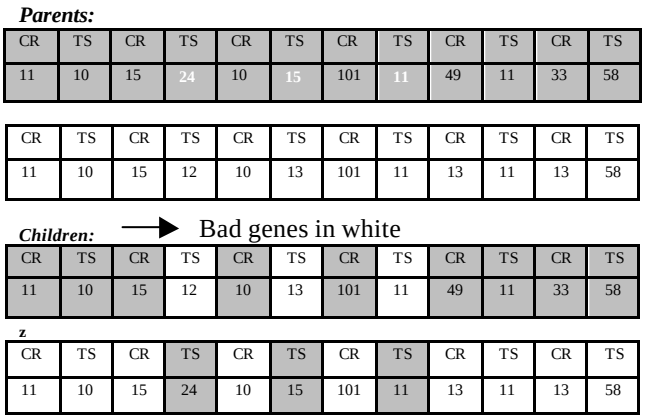

Figure 3 – Bad genes crossover. White alleles refer to bad genes.

2.6.1.2 Bad Gene Mutation

This mutation randomly changes one bad gene of a selected chromosome.

3 Results

3.1 Performance

To test the efficiency of the bad gene operator we used the information of the 96/97 timetables from a large high school called DFL. In Table 2 we can see the characteristics of this school.

Classes	41
Teachers	109
Class-rooms	37
Time slot duration	60 min.
Time slots to be assigned	1311
Number of lessons	472
Rate time slots/teachers	12
Rate time slots /classrooms	69%

Table 2. – Data from the DFL school.

The weights used in the cost function are shown in Table 3. These values lead the algorithm to eliminate the violations of hard constraints first and then to minimise the violations of the soft constraints.

Constraint	Weight
Superimposed lessons/teacher	1.0
Superimposed lessons/ Class	1.0
Superimposed lessons/ classrooms	0.25
Gaps in class timetables	0.008
Gaps in teacher timetables	0.002
Maximum number of lessons per day – class	0.02
Maximum number of consecutive lessons per day – class	0.02
Maximum number of lessons per day – teacher	0.02
Maximum number of consecutive lessons per day – teacher	0.02
Lessons of same subject and type on same day	0.25
Lunch Hour	0.02

Table 3 .– Weights used in the cost function.

3.2 Improvements

In figure 4 we can see the improvement in the E.A. performance caused by the introduction of the bad genes operators. For each configuration, the E.A. was run 15 times for 35000 chromosome evaluations. The results shown in fig. 4 refer to the evolution of the mean values of the cost of the best chromosome. Configuration A refers to an E.A. with the typical monopoint, multipoint and uniform crossover and mutation. Configuration B adds a previous version of the bad gene mutation discussed in [7] to these operators. In this version the mutation is confined to the genes that give rise to the violation of a hard constraint that cannot be repaired. Configuration C is the configuration discussed above with the new bad gene operators with a wider region of infection. In this configuration the bad gene crossover replaced the uniform crossover operator.

The tests were made under the following conditions:

- Population dimension: 40
- Elitism: 5 pass to next generation
- Starting values for the operators' probability: 0.225 for each crossover and one gene mutation per chromosome at a rate of 0.1.

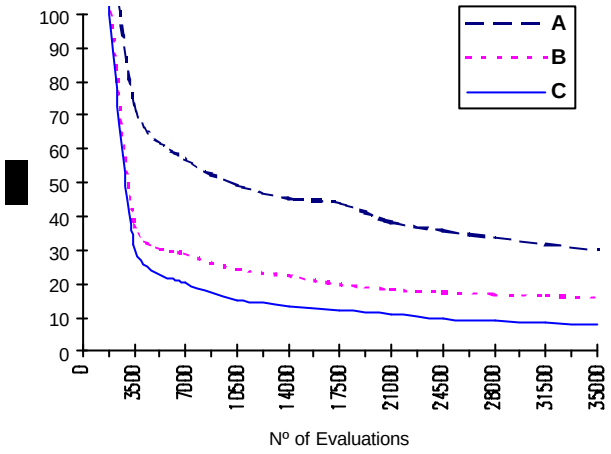

Figure 4 – Evolution of the algorithm with the following configurations:

A – E.A. with Mono-point, multi-point and uniform crossover and mutation.

B – E.A. with previous version of bad gene mutation

C – E.A. with new bad gene operators.

In configurations B and C we increase the convergence speed of the E.A. by making 5 copies of the best chromosomes and applying the bad gene mutation to them. These 5 chromosomes, together with the best 5 of the previous generation (elitism) and the 30 additional chromosomes created by crossover and mutation operators constitute each new generation.

4 Conclusions

The E.A. solves the problem presented satisfying all hard constraints and with very few soft constraint violations. We plan to obtain data from other schools to more thoroughly test the bad gene operators and good results are expected.

We also tested a bad gene crossover with different crossover probabilities for bad genes. In this crossover, the bad genes that actually give rise to the violation of a hard constraint are always swapped and the others are swapped with a 50% probability. This crossover operator has a performance similar to that of configuration B.

The new bad-gene operators result in a faster convergence of the algorithm but are only good for eliminating violations of hard constraints. When none of the hard constraints are violated the operators are useless. We are currently working on new operators that will be especially effective in this case.

References

[1] P. Adamidis, P. Arapakis, Weekly Lecture Timetabling with Genetic Algorithms. *PATAT 97*, 1997, pp 278-280.

[2] J. Allen Lima, J., N. Gracias, H. Pereira, A. C. Rosa, Fitness Function Design for Genetic Algorithms in Cost Evaluation Based Problems, *Proc. IEEE - Int. Conf. Evolutionary Computation, ICEC'96*, 1996, pp 207-212.

[3] J.P. Caldeira, A. C. Rosa, School Timetabling using Genetic Search. *PATAT 97*, 1997, pp 115-122.

[4] T. B. Cooper, J.K. Kingston, The Complexity of Timetable Construction Problems, *Practice and Theory of Automated Timetabling*, Selected Papers, Springer-Verlag, 1995.

[5] L. Davis, *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[6] W. Erben, J. Keppler, A Genetic Algorithm Solving a Weekly Course-Timetabling Problem. *Practice and Theory of Automated Timetabling*, Selected Papers, Springer-Verlag, 1995.

[7] C. Fernandes, J. P. Caldeira, F. Melicio, A. C Rosa, High School Weekly Timetabling by Evolutionary Algorithms, *SAC99*, 1999.

[8] D.E. Goldberg, *Genetic Algorithms in search, optimization and machine Learning*, Addison-Wesley, 1989.

[9] M. Kim, T. Chung, Development of automatic Class Timetable for University. *PATAT 97*, 1997, pp 182-186,.

[10] F. Melicio, J. P. Caldeira, A. C. Rosa, Timetabling implementation aspects by Simulated Annealing. *IEEE-ICSSSE'98*, Beijing, 1998.

[11] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1994.

[12] D. C. Rich, A Smart Genetic Algorithm for University Timetabling. *Practice and Theory of Automated Timetabling*, Selected Papers, Springer-Verlag, 1995

[13] M. Stuber, Real World Timetabling: A Pragmatic View. *PATAT 97*, 1997, pp 258-267.

[14] R. Tavares, A. Teófilo, P. Silva, A. C. Rosa, Infected Genes Evolutionary Algorithm. *SAC99*, 1999.