An Evolutionary Algorithm for Solving the School Time-Tabling Problem

Conference Paper · February 2001

DOI: 10.1007/3-540-45365-2_47 · Source: CiteSeer

2 authors, including:

Andrea G. B. Tettamanzi
University of Nice Sophia Antipolis
250 PUBLICATIONS **2,223** CITATIONS

SEE PROFILE

An Evolutionary Algorithm for solving the School Time-Tabling Problem

Calogero Di Stefano[1] and Andrea G. B. Tettamanzi[2]

[1] Genetica S.r.l.
Via San Dionigi 15, I-20139 Milan, Italy
E-mail: distefano@genetica-soft.com
[2] Università degli Studi di Milano
Dipartimento di Tecnologie dell'Informazione
Via Bramante 65, I-26013 Crema (CR), Italy
E-mail: tettaman@dsi.unimi.it

Abstract. This paper describes an evolutionary algorithm for school time-tabling, demonstrated through applications to the Italian school system. Heuristics have been found and perfected which offer good generalization capabilities. A particular attention has been devoted to problem formulation, also in terms of fuzzy logic, as well as to testing different genetic operators and parameter settings. This work has obtained results of remarkable practical relevance on real-world problem instances illustrated in the paper, and eventually gave rise to a successful commercial product.

1 Introduction

The time-table problem (TTP) is the planning of a number of meetings (e.g., exams, lessons, matches) involving a group of people (e.g., students, teachers, athlets) for a given period and requiring given resources (e.g., rooms, laboratories, sports facilities) according with their availability and respecting some other constraints.

The TTP is an exciting challenge for computational intelligence and operations research, essentially because it is NP-complete [7]. School time-tabling is often even more complicated by the details of a given real situation and real-world problem instances often involve constraints escaping an exact representation, such as constraints related to user personal preferences. Advanced search techniques exploit various heuristics in order to rule out from the search space those regions where an optimum is not expected to exist. One of these heuristics, namely evolutionary algorithms, have been successfully applied to various types of TTP [6, 1, 4] and are among the most promising techniques available to date for solving this kind of problem.

This paper is organized as follows: Section 2 presents the data and the constraints that make up a problem instance and the form of a solution, in order to identify its characterizing aspects. Section 3 describes the evolutionary algorithm implemented. Finally, Section 4 demonstrated the experiments carried out with

the algorithm on some real-world problem instances, and Section 5 discusses the results.

2 The Problem

2.1 Instance Representation

School time-tabling is generally based on a significant volume of data and on different types of constraints. A problem instance is identified by the following entities and relationships:

- rooms, defined by their type, capacity and location;
- subjects, identified by the room type they require and by a priority with respect to other subjects;
- teachers, characterized by the subjects they teach and by their fuzzy availability matrix, defining the degree to which each teacher is available in each period (see Figure 1);

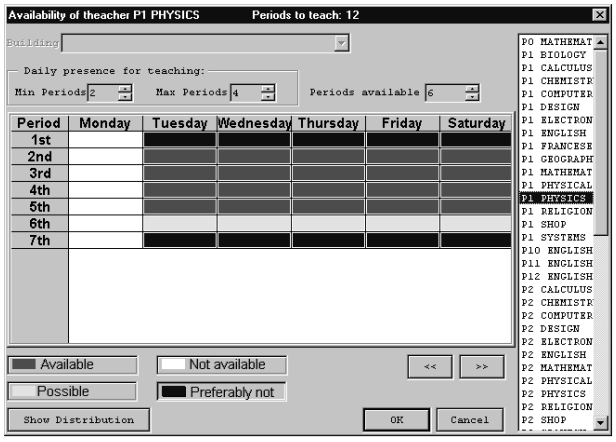

Fig. 1. An example of a fuzzy availability matrix for a teacher.

- classes (i.e., groups of students, including the degenerate case of single students), assigned to a given location (e.g., building) and having their availability matrix, such as the one in Figure 2;
- lessons, meaning the relation $\langle t, s, c, l \rangle$, where t is a teacher, s is the subject, c is the class attending the lesson and l is the duration in periods. More than one teacher and more than one class can participate in a lesson, in which case we speak of *grouping*.

As the reader will have noticed, some constraints of the school TTP are most naturally expressed in fuzzy terms [11].

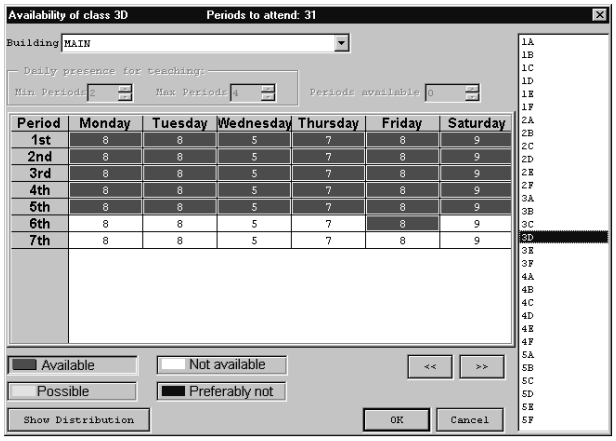

Fig. 2. An availability matrix for a class.

A candidate solution to this TTP is an assignment of a starting period to every lesson.

2.2 Constraints

A candidate solution can be considered satisfactory if it meets the teachers' and students' requirements, and if it respects the availability of resources. These properties are formalized through a number of constraints.

Constraints can be of two types: hard or soft, depending on whether their satisfaction is mandatory or just desirable. Accordingly, a solution will be said to be *feasible* if it satisfies all the hard constraints, and a feasible solution will be said to be more or less *acceptable* depending on the degree to which soft constrains are satisfied.

An important thing to notice is that the classification of each constraint as hard or soft is to some extent arbitrary, and may be thought as being part of the problem instance definition process.

For the school TTP, based on a thorough analysis of the Italian system, the following set of general constraints was identified.

- Constraints on rooms: in each period, the number of rooms used per type cannot be larger than the number of available rooms of that type (*bounded rooms*); rooms of some types are not available for a certain time before or after use for preparation or tidying up (*room availability*).
- Constraints on subjects: some subjects are better taught before others, or earlier in the day when students are fresh, or lessons of some other subjects should not take place after one another, e.g., similar subjects (*priority*).
- Constraints on teachers: a teacher cannot teach more than one lesson at a time (*physical availability*); a teacher must teach a given number of lessons

to each class as provided for by the programme for every subject (*contractual availability*); a teacher should be assigned lessons only in the periods he or she has marked as available (*subjective availability*); if the school has more than one building or location, appropriate time must be allowed for a teacher to travel between buildings when required (*displacement*); finally, teachers prefer that their lessons be concentrated in time, minimizing gaps or isolated lessons (*concentration*).

- Constraints on classes: if the school has more than one location, each class belongs to just one location and classes participating in a lesson must be from the same location (*co-location*); a lesson cannot take place when one of the participating classes is not at school (*presence*); while at school, a class should always be attending a lesson, except for breaks (*no gaps*)—this is a strong requirement in the Italian system, but may not be so important in other systems.
- Constraints on lessons: some lessons must stick to a pre-assigned schedule for organizational reasons (*pre-assignment*); lessons of the same subject should be uniformly distributed over the week (*distribution*); the total weight of lessons for each day should be as uniform as possible over the week (*uniformity*); lesson weight (i.e., priority) should be decreasing within a day (*decreasing burden*); multiple lessons on the same subject, with the same classes on the same day should be scheduled sequentially or, put another way, only a single lesson on a subject is allowed per day for a class (*sequentiality*); subsequent lessons on the same subject must take place in the same room (*locality*); different lessons involving the same teacher or the same class cannot overlap (*non-overlap*).
- Other organizational constraints: for instance, on each day, classes take their lunch break at different hours in order to avoid overcrowding of the lunch room (*break balance*);

The search for an acceptable timetable strongly depends on how constraints are classified. The case where all constraints are soft brings forth an exceedingly broad search space, whereas the opposite case where all constraints are hard is practically meaningless, for it gives rise to a very sparse if not even empty search space.

For a standard problem instance relevant to a typical Italian secondary school, a possible partition of the constraints described above into hard and soft constraints might be the one reported in Table 1.

3 The Evolutionary Algorithm

The approach adopted to solve the problem described in Section 2 is a quite standard elitist evolutionary algorithm, except for the perturbation operator, which alternates between an *intelligent mutation* operator and an *improvement* operator, whose details are given below.

hard constraints		soft constraints	
Constraint	Entity	Constraint	Entity
bounded rooms	rooms	room availability	rooms
physical availability	teachers	priority	subjects
contractual availability	teachers	subjective availability	teachers
co-location	classes	displacement	teachers
presence	classes	concentration	teachers
pre-assignment	lessons	distribution	lessons
sequentiality	lessons	decreasing burden	lessons
no gaps	classes	locality	lessons
non overlap	lessons	uniformity	lessons

Table 1. A possible classification of constraints for the Italian school system.

3.1 Genetic Representation

A *direct* representation has been chosen, along the lines of [4, 1, 2, 5], even though recent work [10] would suggest indirect representation to work much better.

However, given the use of the improvement operator, which is a Lamarckian local search operator, one might argue that our method should not really be classed as having a direct representation. If this operator were always applied then one could possibly describe our algorithm as having an indirect representation with the chromosome providing a parameterisation to the *improvement* operator.

So, in the end, our approach might show the advantages both of an indirect representation and of a direct representation. Moreover, such representation lends itself to specializations of the kinds required by a many-faceted reality like the Italian school system.

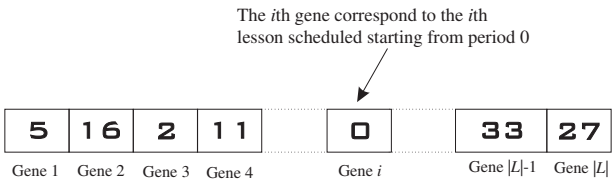

Fig. 3. Representation scheme of a genotype.

An individual in the population is a vector of integers, as depicted in Figure 3, having the same size as the number of lessons to be scheduled. Each integer encodes the starting period of the associated lesson.

The values each gene can take up depends on data like the length of the relevant lesson in periods, the number of days in the school week, the number of periods in each day, and so on.

Other advantages of this direct representation are:

1. a simple procedure for calculating the penalties to assign to the violation of each constraint;
2. easy design of operators which preserve feasibility, thus allowing significant savings of computational power;
3. last but not least, a particularly efficient decoding function which, given a genotype, reconstructs the corresponding timetable.

3.2 Initialization

The population is seeded with genotypes generated by a randomly driven greedy algorithm, which places the lessons one by one by selecting at random a starting period among those that would not violate any constraint. This incremental procedure goes on until the set of "open" starting periods becomes empty, and then places all remaining lessons at random.

The main advantage of this seeding heuristics is that it gives the evolutionary algorithm a jump start, by producing much better timetables than a completely random initialization would do—practically at the same computational cost.

3.3 Fitness

The fitness function employed by our evolutionary algorithm for school timetabling is of the form

$$f(g) = \frac{1}{1 + \sum_i \alpha_i h_i} + \frac{\gamma}{1 + \sum_j \beta_j s_j}, \tag{1}$$

where $h_i \in \mathbb{R}^+$ is the penalty associated to the violation of the ith hard constraint, whose relative weight is $\alpha_i \in \mathbb{R}^+$, and $s_j \in \mathbb{R}^+$ is the penalty associated to the degree of violation of the jth soft constraint, whose relative weight is $\beta_j \in \mathbb{R}^+$; finally, $\gamma \in \{0, 1\}$ is zero until all hard constraints are satisfied, and one since.

The weights attached to each hard and soft constraint allow the algorithm to be fine-tuned to better direct search. In practice, the weights were chosen according to the relative importance of constraints in the problem instances treated. It should be stressed that the results are *very* sensitive to the choice of the weights, because soft constraints correspond to conflicting criteria.

Satisfaction of soft constraints begins only after at least one feasible solution has been found, and γ is switched to one: this device does not let evolution trade off slight violations of hard constraints with high satisfaction of soft constraints.

3.4 Selection

The individuals are chosen for reproduction using tournament selection [3], with tournament size $k \geq 2$, which is a parameter of the algorithm. The implementation of this selection scheme enforces elitism by introducing at least one copy of the best individual into the next generation, without perturbation. Apart from this detail, the algorithm uses a generational replacement reproduction strategy.

3.5 Recombination

Among the cross-over operators described in the literature for an integer representation like the one adopted by our algorithm (see e.g. [9]), the one that proved to be best suited for the TTP, based on experiments on the three sample problem instances discussed in Section 4, was uniform cross-over, whose mechanism is illustrated in Figure 4: this operator treats the integer genes atomically. Recombination is performed on each pair of individuals with probability p_{cross}, which is a parameter of the algorithm.

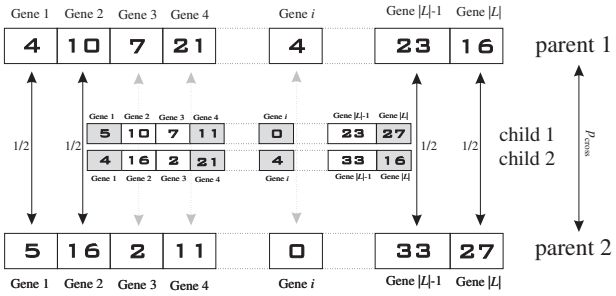

Fig. 4. Illustration of uniform crossover on the genetic representation of a school timetable.

3.6 Perturbation

Typically, mutation in EAs is a blind operator, in the sense that it produces random perturbations, regardless of the fact that they improve or worsen the solution represented by the affected individual.

In contrast with that, the two perturbation operators employed in combination by our algorithm are rather of the Lamarckian type, i.e., based on the law of use and disuse which, if wrong when referred to natural genetics, sometimes leads to more efficient search in artificial evolutionary systems.

The two mutually exclusive perturbation operators are:

– *intelligent mutation*, applied with probability p_{mut}, and

– *improvement*, applied with probability p_{improve}.

At each generation, the decision whether to enable the intelligent mutation or the improvement operator is made by tossing a biased coin. The bias p_{pert}, in favor of intelligent mutation, is a parameter of the algorithm.

Intelligent Mutation This operator, while retaining a random nature, is directed towards performing changes that do not decrease the fitness of the individual. In particular, if the operator affects the ith gene, it will indirectly affect all the other lessons involving the same class, teacher or room. The choice of the "action range" of this operator is random and the variation suffered by the ith gene is such as to make fitness increase. In practice, intelligent mutation aims at reducing constraint violation.

Improvement This operator, which is the perturbing alternative to intelligent mutation, restructures an individual to a major extent.

Restructuring commences by randomly selecting a gene (i.e., a lesson) and concentrates on the partial timetables for the relevant class, teacher and room.

First of all, it aims at making periods for which no lesson has been scheduled contiguous (i.e., compacting "gaps" in the class, teacher, and room timetable). Then it allocates the contiguous free space thus claimed to the the lesson corresponding to the selected gene.

Application of this operator generally results in a fitness increase and preserves the feasibility of a solution. Furthermore, this operator is responsible for injecting substantially novel genetic material into the population.

4 Experiments and results

Experiments have been carried out on a constructed problem instance, designed to test most aspects of the problem. The instance is a realistic model of a generic Italian "industrial technical high-school", which we will call "G. Marconi" for convenience, summarized in Table 4. A total of 942 lessons are to be scheduled.

Further experiments have been carried out on two real state lycées located in Milan, namely the G. Carducci classic state lycée and the Leonardo da Vinci scientific state lycée, both enrolling ca. 1,000 students.

Data for the G. Carducci lycée refer to the school year 1997/1998, with 924 lessons to schedule. Data for the Leonardo da Vinci lycée refer to school year 1996/1997, with 1003 lessons to schedule. Table 4 summarizes information on these two schools.

The parameter settings for the evolutionary algorithm reported in Table 4, are the ones that consistently gave the best results on the three problem instances used for experiments.

A typical run for these problem instances takes 40 minutes on average to come up with a feasible solution and 5 $\frac{1}{2}$ hours to devise an acceptable solution on a 500 MHz PC with 64 Mbyte RAM and the Microsoft NT operating system.

Resources	Main Building	Branch Building	Remarks
classes	12	18	6 sections of 5 classes
teachers	35	27	3 Physical Education teachers work in both buildings
subjects	10	13	6 subjects are taught in both buildings
lessons	558	384	no lesson requires class grouping
labs	8	4	
gyms	2	1	the figures refer to the number classes which can use the gym at one time

Table 2. Summary of the G. Marconi model industrial technical high-school.

Resources	G. Carducci	Leonardo da Vinci
classes	33	35
teachers	60	71
subjects	13	23
lessons	924	1003
labs	1	3
gyms	2	2

Table 3. Summary of the G. Carducci and Leonardo da Vinci lycées.

Parameter	Value
Population size	100
p_{cross}	0.3
p_{pert}	0.5
p_{mut}	0.02
p_{improve}	0.02
Tournament size k	30
α_i, β_j	1 for all i, j

Table 4. Optimal parameter settings for the evolutionary algorithm.

Figure 5 plots the fitness and the penalties associated to constraint violations during a run of the algorithm on the Leonardo lycée. The first feasible solution was found at generation 2,001, and the run was stopped at generation 10,200 when the degree of satisfaction of soft constraints did not look likely to increase anymore.

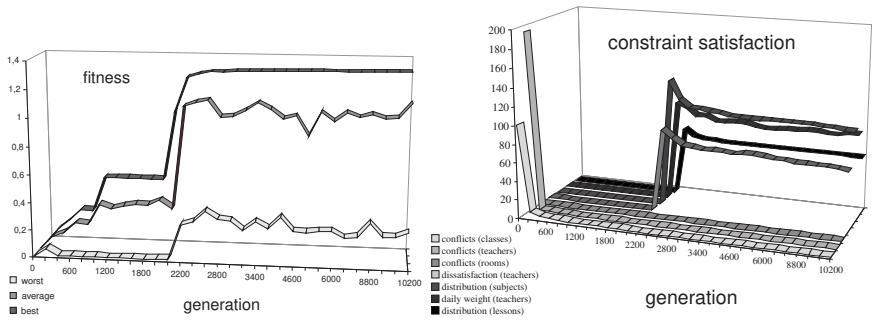

Fig. 5. Fitness graph and evolution of constraint satisfaction during a run for the Leonardo lycée.

5 Discussion

The algorithm described in Section 3 cannot be considered a "pure" evolutionary algorithm; instead, its efficiency and effectiveness rely critically on the hybridization with two heuristics based on local search, implemented by the intelligent mutation and improvement operators.

Even without fine tuning of the algorithm parameters and of the relative constraint weights α_i and β_j, the quality of the results obtained, which cannot be shown here for lack of space, was remarkable, all the more so because no simplifying assumption or constraint elimination had to be made to obtain them.

The timetables obtained were submitted to the timetable committee of the Leonardo da Vinci scientific state lycée, which has been in charge for years of assembling the school timetable at every opening of the school year. Their expert assessment of the results was highly positive and their implementation did not require any manual adjustment.

The practical relevance of the results obtained led Genetica S.r.l. to package the approach decribed in this paper into a full-fledged commercial software product, EvoSchool, which has been successfully launched on the Italian market. Adaptations for other countries are also under way. EvoSchool, version 2.0, is freely available for evaluation at the URL

`http://www.genetica-soft.com/eng/evoschool.html`

The demo comes with the datasets for the three examples described in Section 4.

References

1. P. Adamidis and P. Arakapis. *Weekly lecture timetabling with genetic algorithms.* Proceedings of the 2nd International Conference on the Practice and Theory of Automated Timetabling, University of Toronto, Canada, 1997.
2. A.M. Barham and J.B. Westwood. *A Simple Heuristic to Facilitate Course Timetabling.* J. Opnl. Res. Soc. 29, 1055-1060.
3. A. Brindle. *Genetic algorithms for function optimization.* Technical Report TR81-2, Department of Computer Science, University of Alberta, Edmonton, 1981.
4. J.P. Caldeira and A.C Rosa. *School timetabling using genetic search.* Proceedings of the 2nd International Conference on the Practice and Theory of Automated Timetabling, University of Toronto, Canada, 1997.
5. D. Corne,P. Ross, H. Fang. *Evolutionary Timetabling: Practice, Prospects and Work in Progress.* Presented at the UK Planning and Scheduling SIG Workshop, (Strathclyde, UK, September 1994), organised by P Prosser.
6. W. Erben. *A Grouping Genetic Algorithm for Graph Colouring and Exam Timetabling.* Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling, Constance, Germany, August 16–18, 2000.
7. S. Even, A. Itai, A. Shamir. *On the Complexity of Timetable and Multicommodity Flow Problems.* Siam Journal of Computing, Vol. 5, No.4, December 1976, 691-703.
8. D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, 1989.
9. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, Berlin, 1992.
10. B. Paechter, R. C. Rankin, A. Cumming. *Improving a Lecture Timetabling System for University Wide Use.* Practice and Theory of Automated Timetabling II, Springer-Verlag, LNCS 1408, Berlin, 1998.
11. L.A. Zadeh. *Fuzzy Sets and Applications: Selected Papers.* John Wiley & Sons, New York, 1987.