A Constructive Evolutionary Approach to School Timetabling

Geraldo Ribeiro Filho[1], Luiz Antonio Nogueira Lorena[2]

[1] UMC/INPE - Av Francisco Rodrigues Filho, 399
08773-380 – Mogi das Cruzes – SP Brazil
Phone: 55-11-4791-3743
geraldo@lac.inpe.br

[2] LAC/INPE - Caixa Postal 515
12.201-970 São José dos Campos – SP - Brazil
Phone: 55-12-345-6553
lorena@lac.inpe.br

Abstract. This work presents a constructive approach to the process of fixing a sequence of meetings between teachers and students in a prefixed period of time, satisfying a set of constraints of various types, known as school timetabling problem. The problem is modeled as a bi-objective problem used as a basis to construct feasible assignments of teachers to classes on specified timeslots. A new representation for the timetabling problem is presented. Pairs of teachers and classes are used to form conflict-free clusters for each timeslot. Teacher preferences and the process of avoiding undesirable waiting times between classes are explicitly considered as additional objectives. Computational results over real test problems are presented.

1 Introduction

The timetabling problem consists in fixing a sequence of meetings between teachers and students in a prefixed period of time (typically a week), satisfying a set of constraints of various types. A large number of variants of the timetabling problem have been proposed in the literature, which differ from each other based on the type of institution involved (university or high school) and distinct constraints. A typical timetable instance requires several days of work for a manual solution[1].

Several techniques have been developed to automatically solve the problem[2, 3]. We therefore see algorithms based on integer programming[4], network flow, and others. In addition, the problem has also been tackled by reducing it to a well-studied problem: *graph coloring*[5]. More recently, some approaches based on search techniques appeared in the literature[6]; among others, we have *simulated annealing*[7], *tabu search*[8] and *genetic algorithms*[9, 10, 11].

We consider in this paper a problem known as *school timetabling*: the weekly scheduling for all the classes of a high school, avoiding teachers meeting two classes in the same time, and vice versa. Our main objective was to help administrative staff

of public schools in Brazil. The particular characteristics observed for Brazilian public schools are:

- Full use of available rooms;
- Closed timetabling – at any timeslot all rooms are occupied;
- Usual timeslot conflicts of classes and teachers; and
- Soft constraints for teachers – preferences to some determined timeslots and in general avoiding the waiting timeslots (windows).

Genetic Algorithms (GA) are very well known, having several applications to general optimization and combinatorial optimization problems[12]. GA is based on the controlled evolution of a structured population, and is considered as an *evolutionary algorithm*[13]. The basis of a GA are the recombination operators and the schema formation and propagation over generations. This work presents an application of a Constructive Genetic Algorithm to school timetabling problems.

The Constructive Genetic Algorithm (CGA) is a recently developed approach of Lorena and Furtado [14] that provides some new features to GA, such as a population formed only by schemata, recombination among schemata, dynamic population size, mutation in complete structures, and the possibility of using heuristics in schemata and/or structure representation. Schemata do not consider all the problem data. The schemata are recombined, and they can produce new schemata or structures. New schemata are evaluated and can be added to the population if they satisfy an evolution test. Structures can result from recombination of schemata or complementing of good schemata. A mutation process is applied to structures and the best structure generated so far is kept in the process.

In this work, the school timetabling problem is considered as a clustering problem to be solved using the CGA. Our CGA application presents various new features compared to others GA applications to school timetabling. They include a specific representation for clustering problems, specialized recombination and local search mutation.

2 CGA Modeling

The CGA is proposed to address the problem of evaluating schemata and structures in a common basis. While in other evolutionary algorithms evaluation of individuals is based on a single function (the fitness function), in the CGA this process relies on two functions, mapping the space of structures and schemata onto $\Re_+$.

2.1 Representation

Considering p timeslots in a week, and respecting the lecture requirements of each class, we can form all possible pairs of (teacher, class), which should be implemented in the p timeslots. Let n be the total of possible pairs.

The soft constraints for teachers are considered implicitly encoded in the representation. The set of teachers is partitioned on three levels, according the number of classes and overall time dedicated to the school. All the teachers are asked to identify undesirable timeslots (preference constraints) conformable with their number of classes per week.

Pairs (teacher, class) are represented by binary columns. For example, considering 4 teachers and 5 classes, the column corresponding to the pair (2,3) is

$$a = \begin{matrix} 0 \\ 1 \\ 0 \\ 0 \\ -- \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{matrix} \quad \begin{matrix} \\ \leftarrow \text{ teacher 2} \\ \\ \\ \\ \\ \\ \leftarrow \text{ class 3} \\ \\ \end{matrix}$$

The CGA works over a population of schemata (strings) formed by n symbols, one for each column. For example: s = (#,0,0,0,#,0,1,#,1,0,0,0,1,#,#,0,#,0,1,0,0,0,1,#), is a possible schema. There are three possible symbols:

1	$\rightarrow$	the corresponding column is a *seed* to form a cluster (there is always exactly p seeds inside each schema or structure);
0	$\rightarrow$	the corresponding column is assigned to a cluster; and
#	$\rightarrow$	the column is considered temporarily out of the problem.

The dissimilarity between two columns is then calculated to non-seed columns and all the other columns assigned to a cluster. The result is used to identify the cluster to which non-seed columns will be assigned. The dissimilarity measure between two columns is given by:

$$d_{jk} = 1 - \frac{\sum_i \left| a_i^k - a_i^j \right|}{\sum_i \left(a_i^k + a_i^j \right)} \qquad \textbf{(1)}$$

where:

a_i^k is the value (zero or one) on position i at column k, and

a_i^j is the value (zero or one) on position i at column j.

To find out the cluster to which a non-seed column will be assigned,

- columns are ordered according to the teacher level and the number of preference constraints,
- we take the seed column that is most dissimilar,
- the columns (the non-seed and the chosen seed) are merged into a single one (simple binary OR operation – see *figure 1* for an example) that becomes a new

seed column. The process then continues until all non-seed columns are assigned to a cluster.

0	**0**	0
1	**1**	0
0	**1**	1
0	**0**	0
--	--	--
0	**1**	1
0	**0**	0
1	**1**	0
0	**0**	0
0	**0**	0

Fig. 1. Merging two strings.

After columns to clusters assignments, exactly p clusters $C_1(s), C_2(s), ..., C_p(s)$ are identified, corresponding to the p available timeslots.

2.2 Modeling

Let X be the set of all structures and schemata that can be generated by the *0-1-#* string representation of *section 2.1.*, and consider two functions f and g, defined as $f : X \to \Re_+$ and $g : X \to \Re_+$ such that $f(s_i) \leq g(s_i)$, for all $s_i \in X$. We define the double fitness evaluation of a structure or schema s_i, due to functions f and g, as *fg-fitness*.

The *CGA* optimization problem implements the *fg-fitness* with the following two objectives:

(*interval minimization*) Search for $s_i \in X$ of minimal $\{g(s_i) - f(s_i)\}$, and
(*g maximization*) Search for $s_i \in X$ of maximal $g(s_i)$.

Considering the schema representation, the *fg-fitness* evaluation increases as the number of labels # decreases, and therefore, structures have higher *fg-fitness* evaluation than schemata. To attain these purposes, a problem to be solved using the CGA is modeled as the following *Bi-objective Optimization Problem* (BOP):

$$\begin{aligned} &Min && \{g(s_i) - f(s_i)\} \\ &Max && g(s_i) \end{aligned}$$

$$\text{subj. to} \quad g(s_i) \geq f(s_i) \qquad \forall s_i \in X$$

Functions f and g must be properly identified to represent optimization objectives of the problem at issue. For each schema $s_i \in X$, exactly p clusters

$C_1(s_i), C_2(s_i), ..., C_p(s_i)$ are identified. Functions g and f are defined by $g(s_i) = \sum_{j=1}^{p} \left[\left(|C_j(s_i)| - 1 \right) . |C_j(s_i)| \right] / 2$ and $f(s_i) = g(s_i) - \sum_{j=1}^{p} [\, conflicts(C_j(s_i))]$.

Considering graphs formed by vertices as columns and the edges as possible conflicts between columns (clashes of teachers or classes), function $g(s_i)$ can be interpreted as the total number of possible conflicts in p complete graphs of size $|C_j(s_i)|$. Function $f(s_i)$ decreases this number by the true number of conflicts on the clusters $C_j(s_i)$. When $f(s_i) = g(s_i)$ the p clusters $C_j(s_i)$ are free of conflicts (a possible feasible solution).

3 The Evolution Process

The *BOP* defined above is not directly considered as the set *X* is not completely known. Instead we consider an evolution process to attain the objectives (*interval minimization* and *g maximization*) of the *BOP*. At the beginning of the process, the following two *expected values* are given to these objectives: a non-negative real number $g_{\max} > Max_{s_i \in X} g(s_i)$, that is an upper bound to $g(s_i)$, for each $s_i \in X$; and the interval length $d\, g_{\max}$, obtained from $g_{\max}$ using a real number $0 < d \leq 1$.

Let $g_{\max} = mult \,.\, p \,.\, \left\lfloor \frac{(\lfloor n/p \rfloor - 1) . \lfloor n/p \rfloor}{2} \right\rfloor$. This upper bound is obtained by dividing the number of vertices n in p clusters with approximately the same number of elements (the expression $\lfloor n/p \rfloor$ gives the large integer smaller than n/p), and the same procedure used for $g(s_i)$ is applied, where the positive factor *mult* is considered to certify that $g_{\max} > \underset{s_i \in X}{Max}\, g(s_i)$.

The evolution process is then conducted considering an adaptive rejection threshold, which contemplates both objectives in BOP. Given a parameter $\alpha \geq 0$, the expression

$$g(s_i) - f(s_i) \geq d\, g_{\max} - \alpha \,.\, d[g_{\max} - g(s_i)] \qquad \textbf{(2)}$$

presents a condition for rejection of a schema or structure s_i from the current population.

The right hand side of (2) is the threshold, composed of the expected value to the interval minimization $d\, g_{\max}$, and the measure $[g_{\max} - g(s_i)]$, that shows the difference of $g(s_i)$ and $g_{\max}$ evaluations. For $\alpha = 0$, the expression (2) is equivalent to comparing the interval length obtained by s_i and the expected length

$d\,g_{\max}$. Schemata or structures are discarded if expression (2) is satisfied. When $\alpha>0$, schemata have higher possibility of being discarded than structures, as structures present, in general, smaller differences $[g_{\max} - g(s_i)]$ than schemata.

The *evolution parameter* α is related to time in the evolution process. Considering that the good schemata need to be preserved for recombination, α starts from 0, and then increases slowly, in small time intervals, from generation to generation. The population at the evolution time α, denoted by P_α, is dynamic in size according to the value of the adaptive parameter α, and can be eventually emptied during the process.

The parameter α is isolated in expression (2), yielding the following expression and the corresponding *rank* to s_i: $\alpha \geq \frac{dg_{\max} - [g(s_i) - f(s_i)]}{d[g_{\max} - g(s_i)]} = \delta(s_i)$

At the time they are created, structures and/or schemata receive their corresponding rank value $\delta(s_i)$. This rank is then compared to the current evolution parameter α. At the moment a structure or schema is created, it is then possible to have some figure of its survivability. The higher the value of $\delta(s_i)$, and better is the structure or schema to the BOP, and they also have more surviving and recombination time.

3.1 Selection, Recombination and Mutation

Functions f and g defined in *section 2.2*. drives the evolution process to reach feasible solutions (structures free of conflicts), but the soft constraints are not directly considered. The selection will consider explicitly the soft constraints. Define a new measure $d(s_i) = \frac{[d_1(s_i) + w_{pref}.d_2(s_i) + w_{window}.d_3(s_i)]}{1 + w_{pref} + w_{window}}$, where $d_j(s_i) = [g_j(s_i) - f_j(s_i)]/g_j(s_i)$, j = 1,2,3; w_{pref} is the preference constraint weight, w_{window} is the window constraint weight, and $g_1(s_i) = g(s_i)$, $f_1(s_i) = f(s_i)$ (as defined in section 2.2), $g_2(s_i) =$ number of columns, $f_2(s_i) = g_2(s_i) -$ number of columns with preference in conflict, $g_3(s_i) =$ number of columns, and $f_3(s_i) = g_3(s_i) -$ number of windows.

The population is kept in a non-decreasing order according to the following key: $\Delta(s_i) = (1 + d(s_i))/(n - n_{\#})$, where $n_{\#}$ is the number of # labels in s_i. Schemata with small $n_{\#}$ and/or presenting small $d(s_i)$ are better and appear in first order positions.

The method used for selection takes one schema from the *n* first positions in the population (*base*) and the second schema from the whole population (*guide*). Before recombination, the first schema is complemented to generate a structure representing a feasible solution (all #'s are replaced by 0's). A mutation process is applied to this

complete structure and it is compared to the best solution found so far, which is kept throughout the process. The recombination merges information from both selected schemata, but preserves the number of labels 1 (number of timeslots) in the new generated schema.

Recombination

if $s_{base}(j) = s_{guide}(j)$ then $s_{new}(j) \leftarrow s_{base}(j)$
if $s_{guide}(j)$=# then $s_{new}(j) \leftarrow s_{base}(j)$
if $s_{base}(j)$ = # or 0 and $s_{guide}(j)$=1 then
    $s_{new}(j) \leftarrow 1$ and $s_{new}(i) \leftarrow 0$ for some $s_{new}(i)$=1
if $s_{base}(j)$ = 1 and $s_{guide}(j)$=0 then
    $s_{new}(j) \leftarrow 0$ and $s_{new}(i) \leftarrow 1$ for some $s_{new}(i)$=0

At each generation, exactly *n* new individuals are created by recombination. If a new individual is a schema, it is inserted into the population; otherwise the new individual is a structure, the mutation process is applied, and it is compared to the best solution found so far.

The mutation process has three parts. The purpose of the first two parts is to repair infeasible solutions eventually produced by recombination. The third part maximizes soft constraints satisfaction.

The mutation process can be described as: 1) Class feasibility - for each cluster, while there are repeated classes in this cluster, find in other clusters a missing class on this one, and swap the columns; 2) Teacher feasibility - for each cluster, while there are repeated teachers in this cluster, find in other clusters a missing teacher on this one, for the same class, and swap the columns; and 3) Teacher preference improvement - make columns ordered according to teachers level and number of constraints, and for each column, if the teacher is constrained in the present cluster, find in other clusters a unconstrained teacher which is missing on this cluster and is feasible to swap, and swap the columns.

4 Computational Tests

The computational tests consider four instances, corresponding to two typical Brazilian high schools. Three periods were considered for the *Gabriel* school, respectively, *morning*, *afternoon* and *evening*, and only one period for the *Massaro* school.

When the tests were performed, the schools activities were already begun. The data used in the tests were taken from feasible solutions given by the schools administrative staff. As the teachers precedence levels and preference timeslots were unknown to us, this information were artificially generated. The set of teachers was partitioned into three levels, according to the number of classes and overall time dedicated to the school: teachers giving classes in less than 50% of the all timeslots in the week were considered at level three, between 50% and 75% were considered at level two, and those giving classes in more than 75% of the timeslots, had the precedence level considered one. Teachers in level one precedes the others and so on. The teachers undesirable timeslots (preference constraints) were artificially identified

considering their number of classes per week and the real solution manually obtained by the schools administrative staff.

Tables 1 to 4 show the results for weights (w_{pref} and w_{window}) varying on the set {0, 0.5, 1}. Three runs were made for each weight combination and the average results are reported in the tables lines. The first column resumes the data: number of teachers, classes, timeslots and preference constraints (total and particularized for the teachers in level one). The other columns show the weight values, percentage of preferences attendance (total and for teachers in level one) and number of windows (total and for the teachers in level one) at the best schedule obtained.

It can be seen in the tables that the weights have direct influence on the soft constraints attendance. The percentages of attendance for teacher preferences and final number of windows are comparable to those obtained by manual schedule, aiming the possibility of being in future an important component of administrative school tools.

All tests were made considering the initial population composed of 100 schemata, generated randomly, and considering for each schema, 20% of it's positions filled with zeros, exactly p with ones and #'s in all remaining positions. For each algorithm run the maximum number of generations was set to 60, and 30 new schemata or structures were created at each generation. For each selection, the base schema was taken from the best 33% individuals of the population and the guide schema was taken from the whole population. Computational times reported correspond to a Pentium II 266 MHz machine.

Comparison between the computer generated solutions and real manually obtained solutions was not considered because of the lack of information like teachers preferences timeslots and teachers precedence level, which in practice can be very subjective.

5. Conclusion

The school timetabling problem is very challenging for public schools in Brazil. Several days of work are normally employed to manually solve these problems. We have proposed in this paper a constructive evolutionary approach to school timetabling problems. It considers the usual feasibility problem of teachers and classes allocation avoiding conflicts, and also some soft constraints, like teacher preferences and to avoid waiting times.

The problem was considered as a clustering problem, and adapted to the application of a recently proposed Constructive Genetic Algorithm (CGA). The CGA has been successfully applied to other clustering problems[14]. The weights used at the selection phase may extend the CGA to the class of multicriteria algorithms. The mutation process was highly specialized to this problem. Some algorithm parameter tuning can give even better results. Computational tests with real world instances was promising and the algorithm may result on a useful tool for Brazilian high schools.

Acknowledgments : The second author acknowledges Fundação para o Amparo a Pesquisa no Estado de S. Paulo - FAPESP (proc. 96/04585-6 and 99/06954-7) for partial financial support. The authors acknowledge the referees for their useful suggestions and comments.

Table 1. Results for Gabriel - morning

Gabriel *morning*	w_{pref}	w_{window}	% prefer.	% prefer. (1)	Number of Windows	Number of Windows (1)	Times (sec.)
	0	0	89.39	83.33	55.00	12.33	718.67
	0	0.5	88.33	74.24	33.33	7.33	625.33
30 teachers	0	1	89.85	80.30	33.33	8.67	599.33
17 classes	0.5	0	93.18	83.33	43.00	8.67	687.33
5x5 Timeslots	0.5	0.5	91.52	81.82	36.00	7.33	632.00
220 pref.	0.5	1	90.91	81.82	37.00	10.00	601.33
22 pref. (1)	1	0	93.18	81.82	42.67	11.33	681.00
	1	0.5	92.12	87.88	35.67	7.33	628.00
	1	1	92.88	83.33	36.67	9.67	594.67

Table 2. Results for Gabriel - afternoon

Gabriel *Afternoon*	w_{pref}	w_{window}	% prefer.	% prefer.(1)	Number of Windows	Number of Windows (1)	Times (sec.)
	0	0	92.75	75.76	48.67	6.00	840.33
	0	0.5	92.31	69.70	32.00	3.33	740.00
38 teachers	0	1	93.28	75.76	34.33	3.00	692.00
17 classes	0.5	0	94.52	75.76	49.67	3.33	758.67
5x5 timeslots	0.5	0.5	93.72	83.33	38.67	4.00	687.67
377 pref.	0.5	1	94.16	81.82	35.67	3.00	668.67
22 pref.(1)	1	0	95.05	84.85	51.33	4.33	732.00
	1	0.5	94.16	80.30	41.67	4.33	679.00
	1	1	93.37	77.27	35.67	4.33	648.67

Table 3. Results for Gabriel - evening

Gabriel *evening*	w_{pref}	w_{window}	% prefer.	% prefer. (1)	Number of Windows	Number of Windows (1)	Times (sec.)
	0	0	88.17	75.31	25.00	4.67	574.33
	0	0.5	88.17	76.54	12.33	2.67	518.67
38 teachers	0	1	88.69	79.01	13.00	1.67	503.33
17 classes	0.5	0	90.59	77.78	22.67	3.33	486.33
5x4 timeslots	0.5	0.5	90.24	87.65	15.33	2.00	478.00
386 pref.	0.5	1	89.55	82.72	13.33	2.67	480.33
27 pref.(1)	1	0	90.85	76.54	26.67	2.33	451.00
	1	0.5	90.59	77.78	16.33	3.00	444.67
	1	1	89.90	83.95	16.33	3.00	446.33

Table 4. Results for Massaro

Massaro	w_{pref}	w_{window}	% prefer.	% prefer. (1)	Number of windows	Number of windows (1)	Times (sec.)
	0	0	85.79	66.67	11.33	2.33	182.00
	0	0.5	88.80	86.67	4.67	0.33	169.67
18 teachers	0	1	89.89	76.67	4.00	1.67	163.00
11 classes	0.5	0	93.44	86.67	7.00	1.33	163.67
5x4 timeslots	0.5	0.5	92.62	93.33	4.00	0.67	159.00
122 pref.	0.5	1	93.17	96.67	6.33	1.00	160.00
10 pref. (1)	1	0	93.72	90.00	7.67	1.67	157.33
	1	0.5	93.44	86.67	5.33	1.00	158.33
	1	1	92.90	83.33	6.00	2.33	158.00

References

1. de Werra D.: An introduction to timetabling. European Journal of Operations Research. 19 (1985) 151–162.
2. Carter M. W.; Laporte G.: Recent developments in practical course timetabling. In Carter M. W.; Burke E. K. (eds.) Lecture Notes in Computer Science 1408. Springer-Verlag, Berlin (1998) 3–19.
3. Schaerf, A.: A survey of automated timetabling. Artificial Intelligence Review. n.13 (1999) 87-127.
4. Tripathy, A.: School timetabling : A case in large binary integer linear programming. Management Science, 30 (12) (1984) 1473-1489.
5. Neufeld, G. A.; Tartar, J.: Graph coloring conditions for existence of the solution to the timetabling problem. Communications of the ACM. n. 17 v.8 (1974).
6. Coloni A.; Dorigo, M.; Maniezzo, V.: Metaheuristics for high school timetabling. Computational Optimization and Applications. n. 9 (1998) 275-298.
7. Dowsland K. A.: Simulated annealing solutions for multi-objective scheduling and timetabling. In Modern Heuristic Search Methods. Wiley, Chichester, England (1996) 155–166.
8. Hertz A.: Tabu search for large scale timetabling problems. European Journal of Operations Research, 54 (1991) 39–47.
9. Burke E. K.; Elliman D. G.; Weare R. F.: A hybrid genetic algorithm for highly constrained timetabling problems. In Larry J. Eshelman (ed.) Genetic Algorithms: Proceedings of the 6th In-ternation Conference, San Francisco. Morgan Kaufmann. (1995) 605–610.
10. Burke E. K.; Newall J. P.; Weare R. F.: Initialization strategies and diversity in evolutionary timetabling. Evolutionary Computation 6(1) (1998) 81–103.
11. Corne D.; Ross P.; Fang H.: Fast practical evolutionary time-tabling. In Fogarty T.C. (ed.) Lecture Notes in Computer Science 865. Springer-Verlag, Berlin (1994) 250–263.
12. Holland, J.H.: Adaptation in natural and artificial systems. MIT Press (1975) 11-147.
13. Michalewicz Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer Verlag, Berlin Heidelberg New York (1996).
14. Lorena, L. A N. and Furtado, J. C.: Constructive Genetic Algorithms for Clustering Problems. Evolutionary Computation - to appear (2000). Available from http://www.lac.inpe.br/~lorena/cga/cga_clus.PDF.