A Constructive Evolutionary Approach to School Timetabling

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

Luiz Antonio Nogueira Lorena
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. This constructive type of genetic algorithm considers evaluation of schemata and structures in a common basis. A variable size population is formed only by schemata, considered as building blocks for feasible solutions construction along the generations. A new representation for the timetabling problem is presented. Pairs of teachers and classes are used to form conflict-free clusters for each timeslot. Binary strings representing pairs are grouped based on dissimilarity measurement. Teacher preferences and the process of avoid undesirable waiting times between classes are explicitly considered as additional objectives. Computational results over real test problems are presented.

Keywords: School Timetabling, constructive genetic algorithm, multi-objective heuristics.

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[26].

Several techniques have been developed to automatically solve the problem[2, 4, 7, 22, 27]. Most of the early techniques were based on a simulation of the human way of solving the problem. All such techniques were based on a *successive augmentation*. That is, a partial timetable is extended, lecture by lecture, until all lectures have been scheduled. The underlying idea of all approaches is "schedule the most constrained lecture first", and they differ only on the meaning they give to the expression "most constrained". Later on, researchers started to apply general techniques to this problem. We therefore see algorithms based on integer programming[25], network flow, and others. In addition, the problem has also been tackled by reducing it to a well-studied problem: *graph coloring*[18]. More recently, some approaches based on search techniques

appeared in the literature[8]; among others, we have *simulated annealing*[1, 11], *tabu search*[10, 13, 23, 24] and *genetic algorithms*[3, 5, 6, 9, 21].

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 known, having several applications to general optimization and combinatorial optimization problems[14]. GA is based on the controlled evolution of a structured population, and is considered as an *evolutionary algorithm*[16, 17]. The basis of a GA is 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 approach of Furtado and Lorena [12] 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. They suffer mutation and the best structure generated is kept in the process.

In this work, the school timetabling problem is considered as a clustering problem to be solved using the CGA. A CGA review is presented on section two, describing the representation, modeling, selection, recombination and mutation, and a CGA pseudo-code. Some computational tests were conducted on real world instances.

2. CGA Review

The CGA is proposed to address the problem of evaluating schemata and structures in a common basis. While in the other evolutionary algorithms, evaluations 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 on 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 =

0
1 ← teacher 2
0
0
--
0
0
1 ← class 3
0
0

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,#,1,1,0,0,0,1,#,#,0,#,0,1,0,0,0,1,#), is a possible schema. There are three possible symbols:

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

The dissimilarity between two columns is then calculated to non-seed columns and all the others 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:

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

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 teacher level and number of preference constraints,
- we take the seed column that is most dissimilar,
- the columns (the non-seed and the chosen seed) are collapsed 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

Figure 1: collapsing 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. CGA Modeling

Let C 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 $f : C \to \Re_+$ and $g : C \to \Re_+$ such that $f(s_i) \leq g(s_i)$, for all $s_i \in C$. 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 $s_i \in C$ of minimal $\{g(s_i) - f(s_i)\}$, and
(*g maximization*) Search for $s_i \in C$ 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 Problems* (BOP):

$$\begin{array}{ll} Min & \{g(s_i) - f(s_i)\} \\ Max & g(s_i) \\ \text{subj. to} & g(s_i) \geq f(s_i) \\ & s_i \in C \end{array}$$

Functions f and g must be properly identified to represent optimization objectives of the problems at issue. For each schema $s_i \in C$, 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 $f(s_i) = g(s_i) - \sum_{j=1}^{p} \left[number\,of\,conflicts(C_j(s_i)) \right]$.

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 if 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).

2.3. The Evolution Process

The evolution process in the CGA is conducted to accomplish 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 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

$$g(s_i) - f(s_i) \geq d g_{\max} - \alpha \cdot d[g_{\max} - g(s_k)] \qquad (2.3.1)$$

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

The right hand side of *(2.3.1)* is the threshold, composed of the expected value to the interval minimization $d g_{\max}$, and the measure $[g_{\max} - g(s_k)]$, that shows the difference of $g(s_i)$ and $g_{\max}$ evaluations. For $\alpha = 0$, the expression *(2.3.1)* 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 *(2.3.1)* is satisfied. When $\alpha > 0$, schemata have higher possibility of being discarded than structures, as structures present, in general, smaller differences $[g_{\max} - g(s_k)]$ 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 now isolated in expression *(2.3.1)*, thus yielding the following expression and corresponding *rank* to s_i: $\alpha \geq \dfrac{dg_{\max} - [g(s_i) - f(s_i)]}{d[g_{\max} - g(s_i)]} = d(s_i)$.

At the time they are created, structures and/or schemata receive their corresponding rank value $d(s_i)$. 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 $d(s_i)$, and better is the structure or schema to the BOP, and they also have more surviving and recombination time.

2.4. 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 $d(s_i) = \dfrac{[d_1(s_i) + w_{pref}.d_2(s_i) + w_{window}.d_3(s_i)]}{1 + w_{pref} + w_{window}}$,

where

$d_j(s_i) = \dfrac{[g_j(s_i) - f_j(s_i)]}{g_j(s_i)}$, 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). This complete structure suffers mutation and 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 colors) in the new generated schema.

Recombination

---

if $s_{base}(j) = s_{guide}(j)$ *then* $s_{new}(j) \neg s_{base}(j)$
if $s_{guide}(j)$=# *then* $s_{new}(j) \neg s_{base}(j)$
if $s_{base}(j)$ = # *or* 0 *and* $s_{guide}(j)$=1 *then*
$s_{new}(j) \neg$ 1 *and* $s_{new}(i) \neg$ 0 *for some* $s_{new}(i)$=1
if $s_{base}(j)$ = 1 *and* $s_{guide}(j)$=0 *then*
$s_{new}(j) \neg$ 0 *and* $s_{new}(i) \neg$ 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 new individual is a structure, it suffers mutation, and 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 following pseudo-code describes the mutation process:

Mutation Process

1: *Class feasibility*
For each cluster
While there are repeated classes
Find in other clusters a class missing on this cluster
Swap columns
End_while
End_for
2: *Teacher feasibility*
For each cluster
While there are repeated teachers
Find in other clusters a teacher missing on this cluster for the same class
Swap columns
End_while
End_for
3: *Teacher preference improvement*
Make columns ordered according teacher level and number of constraints
For each column
If the teacher is constrained in the present cluster then
Find in other clusters a unconstrained teacher missing on this cluster
and feasible to swap
Swap columns
End_if
End_for

2.5. The Algorithm

The Constructive Genetic Algorithm can be summed up by the pseudo-code:

CGA

Given g_{max} and d ;
$\alpha := 0$;
$\varepsilon := 0.05$; *{ time interval }*
Initialize P_α ; *{ initial population }*
Evaluate P_α ; *{ fg-fitness }*
For all $s_i \in P_\alpha$ compute $d(s_i)$ { rank computation }
end_for
While (not stop condition) *do*
    For all $s_i \hat{I} P_a$ satisfying $\alpha < d(s_i)$ do *{ evolution test }*
        $\alpha := \alpha + \varepsilon$;
        Select P_α from $P_{\alpha-\varepsilon}$; *{ reproduction operator }*
        Recombine P_α ; *{ recombination operators }*
        Evaluate P_α ; *{ fg-fitness }*
    end_for
    For all new $s_i \in P_\alpha$ compute $d(s_i)$ { rank computation }
    end_for
end_while

The initial population was composed of 100 schemata, generated randomly, considering for each schema, 20% of zeros, exactly p ones and # on the remaining positions.

3. Computational Tests

The computational tests consider four instances, corresponding to two typical Brazilian high schools. Three periods was considered for the *Gabriel* school, respectively, *morning*, *afternoon* and *evening*, and only one period for the *Massaro* school. The set of teachers is partitioned on three levels, according the number of classes and overall time dedicated to the school. Teachers in level one precede the others and so on. All the teachers were asked to identify undesirable timeslots (preference constraints) conformable with their number of classes per week.

Tables 1 to *4* show the results for weights (w_{pref} and w_{window}) varying on the set {0, 0.5, 1}. The 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 timetable. It can be seen in tables that the weights have direct influence on the soft constraints attendance. Computational times reported results from a Pentium II 266 MHz machine. Results can be considered successful aiming the possibility of being in future an important component of administrative school tools.

4. 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 have applied with success to other clustering problems[15, 19, 20]. 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 Conselho Nacional de Desenvolvimento Científico e Tecnológico -CNPq (proc. 350034/91-5, 520844/96-3, 680082/95-6) and Fundação para o Amparo a Pesquisa no Estado de S. Paulo - FAPESP (proc. 95/9522-0 e 96/04585-6) for partial financial support.

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. const.	0.5	1	90.91	81.82	37.00	10.00	601.33
22 pref. const (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 1: Results for Gabriel - morning

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. const.	0.5	1	94.16	81.82	35.67	3.00	668.67
22 pref. const. (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 2: Results for Gabriel - afternoon

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. const.	0.5	1	89.55	82.72	13.33	2.67	480.33
27 pref. const. (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 3: Results for Gabriel - evening

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. const.	0.5	1	93.17	96.67	6.33	1.00	160.00
10 pref. const. (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

Table 4: Results for Massaro.

References

1. Abramson, D.: Constructing schools timetables using simulated annealing: sequential and parallel algorithms. Management Science. n. 37 v. 1. (1991) 98-113

2. Brittan J. N. G.; Farley F. J. M.: College timetable construction by computer. The Computer Journal. 14 (1971) 361–365

3. Burke E .K.; Elliman D. G.; Weare R. F.: Specialized recombinative operators for timetabling problems. In T. C. Fogarty, editor, Lecture Notes in Computer Science 993 (AISB Workshop on Evolutionary Computing). Springer-Verlag, Berlin. (1995) 75-85

4. Burke E. K.; Carter M. (eds.): Practice and Theory of Automated Timetabling II (Lecture Notes in Computer Science1408). Springer-Verlag, Berlin. (1998)

5. 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,

6. Burke E. K.; Newall J. P.; Weare R. F.: Initialization strategies and diversity in evolutionary timetabling. Evolutionary Computation 6(1) (1998) 81–103

7. 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

8. Coloni A.; Dorigo, M.; Maniezzo, V.: Metaheuristics for high school timetabling. Computational Optimization and Applications. n. 9 (1998) 275-298

9. 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

10. Costa, D.: A tabu search algorithm for computing an operational timetable. European Journal of Operational Research. n. 76 (1994) 98-110

11. Dowsland K. A.: Simulated annealing solutions for multi-objective scheduling and timetabling. In Modern Heuristic Search Methods. Wiley, Chichester, England (1996) 155–166

12. Furtado, J. C.; Lorena, L. A N: Constructive Genetic Algorithms for Clustering Problems. Evolutionary Computation – to appear (2000). Available from http://www.lac.inpe.br/~lorena/cga_clus.PDF.

13. Hertz A.: Tabu search for large scale timetabling problems. European Journal of Operations Research, 54 (1991) 39–47,

14. Holland, J.H.: Adaptation in natural and artificial systems. MIT Press (1975) 11-147

15. Lorena, L.A.N. ; Narciso, M. G. and Beasley J. E.: A constructive genetic algorithm for the generalized assignment problem. Evolutionary Optimization. (submitted) (1999)

16. Michalewicz Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer Verlag, Berlin Heidelberg New York (1996)

17. Michalewicz Z.; Fogel D. B.: How to Solve It : Modern Heuristics. Springer Verlag, Berlin Heidelberg New York (2000)

18. 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)

19. Ribeiro Filho, G. and Lorena, L. A. N.: Constructive genetic algorithm and Column Generation: an application to graph coloring. Asian Pacific Journal of Operation Research (APJOR) (submitted) (2000) Available from http://www.lac.inpe.br/~geraldo.

20. Ribeiro Filho, G. and Lorena, L. A. N.: Constructive genetic algorithm for Machine-part Cell Formation. Journal of Intelligent Manufacturing (submitted) (2000) Available from http://www.lac.inpe.br/~geraldo.

21. Ross P.; Corne D.; Fang H-L.: Improving evolutionary timetabling with delta evaluation and directed mutation. In Davidor Y.; Schwefel H-P.; Manner R. (eds.) Parallel Problem Solving in Nature, v.3 Springer-Verlag Berlin (1994) 565–566

22. Schaerf, A.: A survey of automated timetabling. Artificial Intelligence Review. n.13 (1999) 87-127

23. Schaerf, A.: Local search techniques for large high school timetabling problems. IEEE Transactions on Systems Man and Cybernetics Part A – Systems and Humans. n.29 (1999) 368-377

24. Schaerf, A.: Tabu search techniques for large high school timetabling problems. In: XIII National Conference on Artificial Intelligence Proceedings. (1996) 363-368

25. Tripathy, A.: School timetabling : A case in large binary integer linear programming. Management Science, 30 (12) (1984) 1473-1489

26. de Werra D.: An introduction to timetabling. European Journal of Operations Research. 19 (1985) 151–162

27. de Werra D.: Some combinatorial models for course scheduling. Lecture Notes in Computer Science, 1153. Springer-Verlag, Berlin (1996) 296–308