Chapter 1

A GRASP-TABU SEARCH ALGORITHM FOR SOLVING SCHOOL TIMETABLING PROBLEMS

Marcone Jamilson Freitas Souza
Department of Computer Science
Federal University of Ouro Preto, Brazil
marcone@iceb.ufop.br

Nelson Maculan
Systems Engineering and Computer Science Program
Federal University of Rio de Janeiro, Brazil
maculan@cos.ufrj.br

Luis Satoru Ochi
Department of Computer Science
Fluminense Federal University, Brazil
satoru@dcc.ic.uff.br

Abstract This work proposes a hybrid approach to solve school timetabling problems. This approach is a GRASP algorithm that uses a partially greedy procedure to construct an initial solution and attempts to improve the constructed solution using a Tabu Search algorithm. When an infeasible solution without overlapping classes is generated, a procedure called Intraclasses-Interclasses is activated, trying to retrieve feasibility. If successful, it will be reactivated. This time, attempting to improve the timetable's compactness as well as other requirements. Computational results show that the Intraclasses-Interclasses procedure speeds up the process of obtaining better quality solutions.

Keywords: Metaheuristics, GRASP, Tabu Search, School Timetabling

1. Introduction

The school timetabling problem (STP) regards the weekly class schedule. The problem consists of coordenating lessons with periods satisfying a set of requirements. The difficulty of solving a real case is well known, since the problem has a huge search space and is highly constrained. As it is a NP-hard problem (Even et al. 1976), a heuristic approach to solve it is justified.

Among the recent techniques that have been successfully used to solve the problem, the following metaheuristics techniques stand out: Tabu Search (Schaerf 1996, Alvarez-Valdes et al. 1996, Costa 1994), Simulated Annealing (Abramson 1991), Genetic Algorithms (Carrasco and Pato 2001, Colorni et al. 1998) etc.

In this work, a local search technique, called Intraclasses-Interclasses (II), is developed. It is introduced in the course of the local search phase of GRASP (Greedy Randomized Adaptive Search Procedures) (Feo and Resende 1995) to try speed up the obtaining of better quality solutions. Procedure II attempts to improve a given timetable without overlapping classes in 2 steps. First, it tries to retrieve feasibility and if successful, it is reactivated. This time, it seeks to improve timetabling quality requirements. This procedure can be applied to solve school timetabling problems, as described in Section 1.2, in which the number of lessons and the periods set aside for classes are the same.

This paper is organized as follows. In Section 1.2, the problem in question is described; the following sections regard the used representation, the neighborhood structure and the objective function. Section 1.5 describes, in detail, the proposed algorithm, and Section 1.6 presents some experimental results. The last Section concludes this investigation.

2. Problem Description

The school timetabling problem (STP) in question consists of a set of m teachers, n classes, s subjects and p weekly periods which are set aside for classes. Periods are distributed in d week days and h daily periods, which occur during the same shift, i.e., $p = d * h$. Classes, which are always available, are disjoint sets of students who take the same subjects. Each subject of a given class is associated with only one teacher, who is previously determined. Furthermore, the number of weekly lessons for each class is precisely p. The following requirements must be satisfied:

(a) a teacher cannot teach more than one class at the same time, i.e, a timetable cannot have an overlapping of teachers;

(b) a class cannot have a lesson with more than one teacher at the same period, i.e, a timetable cannot have an overlapping of classes;

(c) each teacher must fulfill his/her weekly number of lessons;

(d) a teacher cannot be scheduled for a period in which he/she is not available;

(e) a class cannot have more than two lessons a day with the same teacher;

(f) teachers' request for double lessons (lessons conducted in two consecutive periods) should be granted as often as possible;

(g) teacher's agenda should be as compact as possible.

Definition 1 A timetable Q that does not satisfy at least one of the requirements (a),. . . ,(e), is considered infeasible and cannot be implemented by the school.

Definition 2 A timetable Q is considered to be type-1 infeasible if one of the following conditions is satisfied in at least one period: (i) There is a class having a lesson with more than one teacher (constraint (b) is not verified); (ii) There is a class having no lesson.

Definition 3 A timetable Q is considered to be type-2 infeasible if constraint (e) is not satisfied by a teacher.

3. Problem Representation and Neighborhood Structure

Teachers' timetable is represented as a matrix $Q_{m\times p}$ of integer values. Each line i in Q represents the weekly schedule for teacher i. Each element $q_{ik} \in \{-1, 0, 1, 2, \ldots, n\}$ indicates the activity of teacher i in period k. Positive values represent the classes of the teachers in the period. Negative values indicate the teacher is unavailable, whereas null values indicate inactivity in the period.

A neighbor of a timetable Q is a timetable Q' that can be reached from Q through a movement consisting of a mere change of two distinct and non-negative values of a given line of Q. This movement is identified by the triplet $< i,\ k,\ \bar{k} >$, where k and $\bar{k}$ represent the periods in which the activities q_{ik} and $q_{i\bar{k}}$ of teacher i will be interchanged.

One can observe that this kind of movement may produce type-1 or type-2 infeasibility. Nevertheless, the possibility of having a teacher that

teaches more than one class at the same time (violation of constraint (a)) is automatically rejected by this representation. In Section 1.5.1 we note that constraint (c) is always satisfied by the representation. The constraint (d) is also always satisfied because movements involving non-negative values are not permitted.

4. The Objective Function

A timetable Q is evaluated according to the following objective function, based on penalties, and should be minimized:

$$f(Q) = \omega * f_1(Q) + \delta * f_2(Q) + \rho * f_3(Q) \tag{1.1}$$

The first two components respectively measure, type-1 and type-2 infeasibility levels, and the third one measures the level of satisfaction regarding the granting of teachers' requests. The weights ω, δ and ρ are chosen to satisfy the condition: $\omega > \delta \gg \rho$ and according to Definition 1.2, Q is feasible if $f_1(Q) = f_2(Q) = 0$.

Type-1 infeasibility level of Q, $f_1(Q)$, is measured by adding to each period k: (a) the number of times l_k a class has no activity in k; (b) the number of times s_k more than one teacher teaches the same class in period k.

Regarding type-2 infeasibility, timetable Q is evaluated by adding the number of times e_i constraint (e) is not satisfied by each teacher i.

The satisfaction of the granting of teachers' requests is measured with relation to the compactness of the timetable (constraint (g)) as well as the compliance with the number of double lessons required (constraint (f)). More precisely,

$$f_3(Q) = \sum_{i=1}^{m} (\alpha_i * b_i + \beta_i * v_i + \gamma_i * c_i) \tag{1.2}$$

where α_i, β_i and γ_i are weights that reflect, respectively, the relative importance of the number of "holes" b_i (periods of no activity between two lesson periods during the same day), the number of week days v_i each teacher is involved in any teaching activity during the same week, and the non-negative difference c_i between the minimum required number of double lessons and the effective number of double lessons in the current agenda of each teacher i. We note that the definition of b_i implies that no difference is made between a hole of two periods and two holes of one period.

5. The Algorithm

The proposed algorithm, called GTS-II, is a GRASP heuristic (Feo and Resende 1995) in which an initial solution is generated by a partially greedy constructive procedure (see Section 1.5.1). Refinement is obtained by means of a Tabu Search method (Section 1.5.2). When a solution without type-1 infeasibility is generated, the TS method activates the Intraclasses-Interclasses procedure (see Section 1.5.3). This construction and refinement sequence is repeated by $GTSmax$ iterations. The best overall solution is kept as the result. The pseudo-code of GTS-II algorithm for minimization is presented in Figure 1.1.

```
procedure GTS-II()
 1   Let Q* be the best timetable and f* its value;
 2   f* ← ∞;
 3   GTSmax ← Maximum number of GTS-II iterations;
 4   for (Iter = 1, 2, ..., GTSmax) do
 5      Q^0 ← grasp-construction();
 6      Q ← TS-II(Q^0);
 7      if (f(Q) < f*) then
 8         Q* ← Q;
 9         f* ← f(Q);
10      end-if;
11   end-for;
12   Return Q*;
end GTS-II;
```

Figure 1.1. Pseudo-code of GTS-II Algorithm

5.1 Generating an initial solution

An initial solution is generated in a constructive way by means of a partially greedy procedure, according to the description that follows.

At first, the periods are sorted according to the number of available teachers, that is, the periods with smallest number of available teachers are on the top. Next, the unscheduled lessons are sorted according to the activity degree of each teacher, that is, the lessons, whose teachers have a larger number of lessons and unavailable periods, have priority. Then, well-ranked lessons are placed in a restricted candidate list (RCL) and a lesson is selected randomly from the RCL. Next, using the critical period order, the selected lesson is scheduled, so that there is no type of infeasibility (in this case, violation to constraints (b) and (e)). In the

event this is impossible, only violation to constraint (e) is admitted at this time. If the impossibility still persists, the lesson will be scheduled by admitting violation to constraint (b) as well. Every time a lesson is scheduled, critical periods are updated as well as the list of remaining unscheduled lessons.

Since all lessons will be scheduled, even though some sort of infeasibility may occur, constraint (c) is automatically guaranteed.

5.2 Tabu Search

Starting from an initial solution generated by the constructive procedure, Tabu Search (TS) metaheuristic follows iteratively exploring the whole neighborhood $N(Q)$ of the current solution Q by means of movements defined in Section 1.3 and guided by the objective function described in (1.1). Then, the algorithm goes to neighbor Q', which produces the smallest $f(Q')$ value, regardless of being worse than the current $f(Q)$ value.

In order to prevent the occurrence of cycling, each time a movement is done, it is stored in a tabu list T. Such list contains the $|T|$ most recent movements and reduces the risk of revisiting one of the $|T| - 1$ last solutions previously visited. Since the tabu list can be very restrictive (Glover and Laguna 1997), the TS algorithm also uses an aspiration criteria by objective. In this way, a movement loses its tabu status if it produces a solution Q' whose value is smaller than the best solution value, $Q^\star$, that had been obtained so far.

Whenever a timetable without overlapping classes is generated (i.e., $f_1(Q) = 0$), the Intraclasses-Interclasses procedure is activated (lines 12 to 15 in Figure 1.2) if no solution with the same value has yet been submitted to II. Reverse movements to those made by II procedure are introduced into the tabu list. The TS procedure continues the search from a solution produced by II and is performed for $TSmax$ iterations without improving the value of the best solution reached so far.

The pseudo-code of TS algorithm using the procedure II is presented in Figure 1.2.

5.3 Intraclasses-Interclasses Procedure

The Intraclasses-Interclasses procedure (II) is based on shortest paths. This procedure is activated when a solution without type-1 infeasibility is available. First, II attempts to make the f_2 component equal to zero. Should it succeed, improvement of the f_3 component, respecting constraints (a) through (e), is attempted, i.e., considering only feasible solutions. Alvarez-Valdes et al. (1996) use a similar procedure, but only to

```
procedure TS-II(Q)
 1   Let Q be a initial timetable;
 2   Q* ← Q;                          /* Best timetable reached so far */
 3   Iter ← 0;                                  /* Iteration counter */
 4   BestIter ← 0;   /* Iteration at which Q* has been found */
 5   T ← ∅;                                              /* Tabu List */
 6   TSmax ← Maximum number of consecutive iterations
                 without improving f(Q*);
 7   while (Iter − BestIter < TSmax) do
 8     Iter ← Iter + 1;
 9     Let Q' ← Q ⊕ m the best neighbor in N(Q) such that
          either the move m does not be tabu (m ∉ T) or
          Q' satisfies the aspiration criteria (f(Q') < f(Q*));
10     Update the tabu list T;
11     Q ← Q';
12     if (f1(Q') = 0) then
13       Q ← Intraclasses-Interclasses(Q');
14       Update the tabu list T;
15     end-if;
16     if (f(Q) < f(Q*)) then
17       Q* ← Q;
18       BestIter ← Iter;
19     end-if;
20   end-while;
21   Return Q*;
end TS-II;
```

Figure 1.2. Pseudo-code of Tabu Search procedure using II

improve a feasible timetable. The type of movement under consideration in their study is also more restricted than II's.

Since II acts similarly, either to retrieve feasibility or to improve a timetable, only the working principles for the second case will be presented.

Let us assume that a solution with no infeasibility is available. Thus, given a timetable Q under these conditions ($f_1(Q) = f_2(Q) = 0$), the graph of class j is defined by $G_j = (V_j, A_j)$, where V_j is the set of periods reserved for class j. A_j is a set of oriented arcs, and is defined as follows: $A_j = \{(k, \bar{k}) :$ the teacher who will teach to class j in period k is available in period $\bar{k}$ and requirement (e) in Section 1.2 is respected in period $\bar{k}\}$.

To each arc $(k, \bar{k}) \in G_j$, a cost $\Delta f_i(k, \bar{k})$ is associated. It represents the cost variation of transferring teacher i from period k to period $\bar{k}$, taking only the f_3 component of the objective function into consideration. Thus, the cost is obtained by calculating the difference between the values of the objective function, regarding the teacher, in the old and new configurations, i.e.:

$$\Delta f_i(k, \bar{k}) = f_i(\bar{k}) - f_i(k) \tag{1.3}$$

where $f(.) = (\rho * f_3)(.)$

Table 1.1 shows a fragment of a timetable. Each line i represents a teacher ($i = T_1, T_2, T_3, T_4$) and each column k represents a period ($k = P_1, P_2, P_3, P_4, P_5$) of the same day. Each element (i, k) in this table represents the activity of teacher i in period k. A, B, C and D are classes. A dash (–) means that the teacher is unavailable, whereas an empty cell indicates there is no activity in the period. Column f_i indicates the value of the objective function of each teacher. It is defined in (1.1) and (1.2) taking $\rho = 1$, $\alpha_i = 1$ and $\beta_i = \gamma_i = 0$ $\forall i$, that is, only holes in the schedule are relevant. The cost of this timetable is $f(Q_1) = f_{T_1} + f_{T_2} + f_{T_3} + f_{T_4} = 1 + 1 + 0 + 0 = 2$.

Table 1.1. Timetable Q_1

	P_1	P_2	P_3	P_4	P_5	f_i
T_1	A		B	B		1
T_2	B	C		A	A	1
T_3		B	A	C	B	0
T_4	C	A	C	D	–	0

Figure 1.3 represents class A graph, G_A. Each period is represented by a vertex to which a teacher is associated. The -1 cost arc (P_1, P_5) indicates that in a case where teacher T_1 changes his/her lesson from period P_1 to period P_5, the value of the objective function will be reduced by 1 unit ($\Delta f_{T_1}(P_1, P_5) = f_{T_1}(P_5) - f_{T_1}(P_1) = 0 - 1 = -1$).

In order to find a timetable that presents a smaller value for the objective function, one should only search for a negative cost cycle in G_j. In the example under consideration, the arc sequence $\{(P_1, P_5), (P_5, P_3), (P_3, P_1)\}$ forms a cycle that has a total cost of -1 $(= -1 + (-1) + 1)$. This sequence defines a set of *intraclasses* movements.

After updating graph G_A and timetable Q_1 with these movements, it is necessary to check once again whether there are negative cost cycles

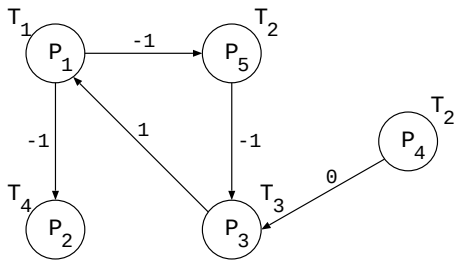

Figure 1.3. G_A, Class A Graph

in the class A graph. In the event that there are no such cycles, the idea is, then, to repeat the procedure with another class and so forth, until it is no longer possible to improve the teachers' timetable by means of intraclasses movements.

Nevertheless, the existence of a negative cost cycle cannot guarantee a better value of the objective function. Moreover, it can generate type-2 infeasible solutions, as demonstrated by Souza (2000). However, such situations only happen when the same teacher is associated to more than one vertex in a cycle. Thus, it is necessary to check the feasibility and the value of the objective function after candidate movements. In order to find other negative cost cycles while using a similar type graph, one should proceed as follows: Choose any arc from the cycle, $(k, \overline{k}) \in G_j$, and insert it in a forbidden movement list L. Then, update the graph of the class under evaluation, excluding the arcs belonging to L from G_j, and search for another negative cost cycle. When it is no longer possible to find negative cost cycles, one should move to another class and eliminate list L.

At the end of the Intraclasses procedure, negative cost arcs may still remain in the graphs of the classes. Figure 1.4, which considers the graphs of classes j and $\bar{\jmath}$, illustrates this situation.

In Figure 1.4, there is a negative cost arc in class j from k to $\bar{k}$, i.e, $\Delta f_i^j(k, \bar{k})$. This indicates that the value of the objective function is likely to be improved, if the lesson assigned to teacher i in period k is transferred to period $\bar{k}$. However, such change cannot occur, because teacher $\bar{\imath}$, who teaches in period $\bar{k}$, is unavailable in period k (he is teaching class $\bar{\jmath}$). The idea is to switch teacher's $\bar{\imath}$ lesson periods, so as to enable the search for a negative cost cycle, involving the graphs of both classes, that are connected to periods k and $\bar{k}$. Let $c^{\bar{\jmath}}(\bar{k}, k)$ be the cost of the shortest path from $\bar{k}$ to k in $G_{\bar{\jmath}}$. The existence of a negative cost cycle involving both classes may be verified by checking whether the condition $\Delta f_i^j(k, \bar{k}) + c^{\bar{\jmath}}(\bar{k}, k) < 0$ is satisfied while transferring teacher $\bar{\imath}$

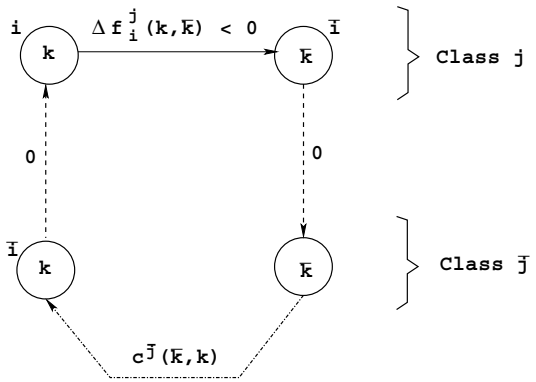

Figure 1.4. Graphs of j and $\bar{j}$ classes after Intraclasses procedure

from class j to class $\bar{j}$ in period $\bar{k}$ and from class $\bar{j}$ to class j in period k. It is noted that there are no costs involved in moving one teacher from one class to another in the same period (null cost arcs in Figure 1.4).

Thus, to each negative cost arc $(k, \bar{k}) \in G_j$, one should investigate whether there is a negative cost cycle involving this arc of G_j, the graph $G_{\bar{j}}$ and the null cost arcs that connect them. This arc sequence defines the so-called *interclasses* movements. Like the intraclasses movements, one must check the feasibility and the value of the objective function after candidate movements.

The Intraclasses-Interclasses procedure is therefore composed of two steps. First, the Intraclasses procedure is applied, resulting in n graphs G_j, $\forall j = 1, \ldots, n$ and, possibly, negative cost arcs. When the first step is finished, Interclasses procedure is applied.

When II procedure is activated, in an attempt to retrieve the feasibility of a solution which has only type-2 infeasibility, the cost of an arc (equation 1.3) is evaluated by assuming that $f(.) = f_2(.)$ Once this attempt is finished, the arc cost is evaluated again based on the cost variation of function (1.1).

Figure 1.5 illustrates how Intraclasses-Interclasses procedure works.

6. Experimental Results

We have evaluated the performance of the GTS-II Algorithm using data of a Brazilian public high school, Escola Dom Silvério, located in Mariana, Minas Gerais State. The requirements for a timetable in this school are displayed in Section 1.2.

The GTS-II algorithm was implemented in C language and tested on a Pentium II PC (350 MHz, 64 MB RAM) running Linux operating

procedure II(Q)
1 Let Q be a initial timetable satisfying $f_1(Q) = 0$,
that is, Q is not type-1 infeasible;
/* If timetable Q is type-2 infeasible call Intraclasses
and Interclasses to repair the infeasibility, that is,
these procedures will use f_2 as objective function /*
2 if $(f_2(Q) \neq 0)$ then
3 $Q \leftarrow$ Intraclasses(Q);
4 if $(f_2(Q) \neq 0)$ then $Q \leftarrow$ Interclasses(Q);
5 end-if;
/* If timetable Q is not type-2 infeasible
call Intraclasses and Interclasses to improve
teachers' personal requests, that is,
these procedures will use f_3 as objective function /*
6 if $(f_2(Q) = 0)$ then
7 $Q \leftarrow$ Intraclasses(Q);
8 if $(f_2(Q) \neq 0)$ then $Q \leftarrow$ Interclasses(Q);
9 end-if;
10 Return Q;
end II;

Figure 1.5. Pseudo-code of Intraclasses-Interclasses procedure

system. In order to determine negative cost cycles, the *Floyd* algorithm (Ahuja et al. 1993) was implemented.

In order to test the efficiency of the II procedure, the GTS-II algorithm was compared to the GTS algorithm, which does not include such procedure (In GTS there are not the lines 12 to 15 in Figure 1.2). The following parameters were set, taking into consideration that $\#lessons$ represent the number of remaining lessons to be scheduled: $|RCL| = \max\{1, (\#lessons)/10\}$, $TSmax = 500$, $GTSmax = 1$, $\omega = 100$, $\delta = 30$, $\rho = 1$, $\alpha_i = 3$, $\beta_i = 9$ and $\gamma_i = 1 \; \forall \, i = 1, ..., m$.

Table 1.2 presents a few characteristics of the data sets and summarizes the results. Columns m, n and $\#lessons$ represent, respectively, the number of teachers, classes and lessons to be scheduled. Column *Sparseness ratio* indicates the sparseness ratios of the data sets, that is, $Sparseness = \frac{m \times p - (\#lessons + u)}{m \times p}$, where u represents the number of unavailable periods and $p = 25$ the weekly periods which are set aside for classes. Lower values indicate a harder problem. The results of Table 1.2 are based on 25 runs, each one initialized by a different seed of random numbers. However, each run had the same seed for both algorithms. To

each 25 run set, the average best solution $\lceil f^\star \rceil$ and the average CPU time (minute:second) are shown.

Table 1.2. Computational Results

				Sparseness	GTS		GTS-II	
Data sets	m	n	*#lessons*	*ratio*	$\lceil f^\star \rceil$	CPU time	$\lceil f^\star \rceil$	CPU time
DS00A	14	6	150	0.50	368	01:53	356	01:41
BR89M	16	8	200	0.30	491	02:17	481	01:24
DS00M	23	12	300	0.13	749	06:01	741	06:02
DS98M	31	13	325	0.58	831	15:36	826	08:09
DS00N	30	14	350	0.52	847	16:10	710	09:40
DS00D	33	20	500	0.39	1164	38:45	1101	29:00

Figure 1.6 illustrates a typical evolution of the best solution in the early seconds of a GRASP iteration in GTS and GTS-II algorithms. As it can be observed, better quality solutions are obtained faster when II is used.

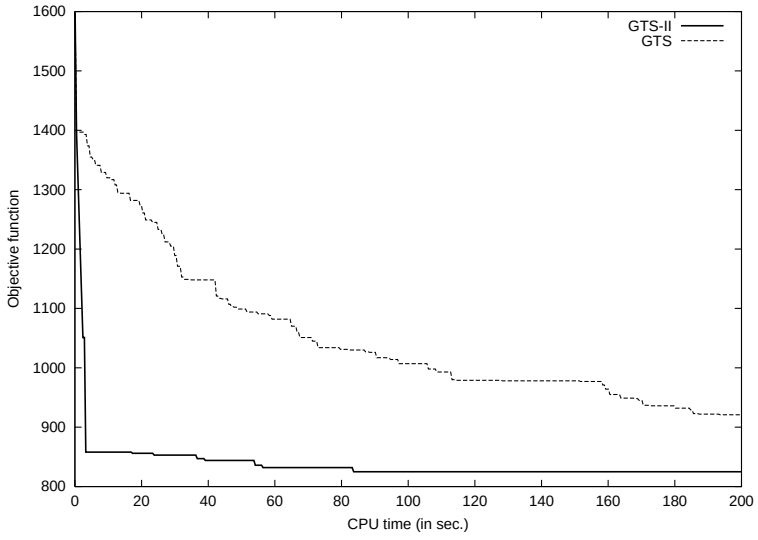

Figure 1.6. Typical performance of the GTS and GTS-II algorithms

7. Conclusions

A hybrid metaheuristic is developed for solving STP. While the partially greedy constructive procedure generates good initial solutions and diversifies the search, the Tabu Search procedure refines the search.

A contribution of our study is the development of the Intraclasses-Interclasses procedure. It is applied to intensify the search when TS generates a timetable without overlapping classes. In this situation, if the solution has some other type of infeasibility, it will try to retrieve feasibility. If successful, it will be applied again. This time, it will try to improve the compactness of the timetable as well as the other requirements.

Finally, besides the requirements mentioned in Section 1.2, the GTS-II algorithm can also be applied to treat other requirements such as: a daily limit of lessons per teacher, teachers' period preference, interval between lessons of the same subject to the same class etc. In such cases, one should simply add the components which measure the difference between the current and the desired solution to the objective function.

Acknowledgment

This work was partially supported by CAPES, Brazil.

References

D. Abramson. Constructing school timetables using simulated annealing: sequential and parallel algorithms. *Management Science*, 37:98–113, 1991.

R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, New Jersey, 1993.

R. Alvarez-Valdes, G. Martin, and J.M. Tamarit. Constructing good solutions for the spanish school timetabling problem. *Journal of the Operational Research Society*, 47:1203–1215, 1996.

M.P. Carrasco and M.V. Pato. A multiobjective genetic algorithm for the class/teacher timetabling problem. In E.K. Burke and W. Erben, editors, *Practice and Theory of Automated Timetabling III*, volume 2079 of *Lecture Notes in Computer Science*, pages 3–17. Springer-Verlag, Konstanz, Germany, 2001.

A. Colorni, M. Dorigo, and V. Maniezzo. Metaheuristics for high school timetabling. *Computational Optimization and Applications*, 9: 275–298, 1998.

D. Costa. A tabu search algorithm for computing an operational timetable. *European Journal of Operational Research*, 76:98–110, 1994.

S. Even, A. Itai, and A. Shamir. On the complexity of timetabling and multicommodity flow problems. *SIAM Journal of Computation*, 5:691–703, 1976.

T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Boston, 1997.

A. Schaerf. Tabu search techniques for large high-school timetabling problems. In *Proceedings of the 30th National Conference on Artificial Intelligence*, pages 363–368, 1996.

M.J.F. Souza. *School timetabling: an approximation by metaheuristics (in portuguese)*. Phd thesis, Systems Engineering and Computer Science Program, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil, December 2000.