A GRASP Strategy for a More Constrained School Timetabling Problem

Arnaldo Vieira Moura
Rafael Augusto Scaraficci

Technical Report - IC-07-006 - Relatório Técnico

February - 2007 - Fevereiro

A GRASP Strategy for a More Constrained School Timetabling Problem

Arnaldo Vieira Moura [*] Rafael Augusto Scaraficci [†]

Abstract

This work treats a typical Brazilian school timetabling problem, that consists of scheduling a set of lectures and teachers in a prefixed period of time, satisfying a set of operational requirements. We applied a basic GRASP heuristic, followed by a path-relinking improvement. The algorithms use a local search procedure that interleaves two types of movements and a path-relinking strategy that is enhanced with a local search procedure. They were tested with real instances and proved to be good approaches to treat this problem. Although some restrictions are specific to the Brazilian educational institutions, the same ideas can inspire similar approaches for solving the school timetabling problem in other situations.

1 Introduction

In educational institutions, we can distinguish three different types of timetabling problems: the school timetabling, the course timetabling and the examination timetabling. Each of these problems has a proper focus that varies according to the country and educational institution [Sch99].

These problems are very difficult [EIS76] and a considerable attention has been devoted to them since the 60's, with the pioneer work of Gotlieb [Got63]. Most of the early techniques were based on the human way to solve the school timetabling problem (STP), where a timetable is constructed scheduling lecture by lecture [SS80]. Later on, researchers started to apply more general techniques, developing algorithms based on integer programming [Tri84], network flow [OdW82] and graph coloring [NT74]. More recently, successful approaches appeared that use metaheuristics, more proeminently those algorithms based on simulated annealing [AKD99], tabu search [SMO03, Sch96], genetic algorithms [Fan94] and GRASP [SMO03].

Among the different educational timetabling problems, our work focus on the school timetabling problem, which consists of scheduling a sequence of lectures involving teachers and classes in a prefixed period of time (typically a week), satisfying a set of operational

[*]Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas.

[†]Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP. Research supported by FAPESP.

requirements. In Brazil, this problem is usually solved manually, requiring expert professionals and many hours of work. In some cases, more than one week is needed in order to construct a feasible school timetable

This work is singular because it treats a set of constraints that are specific to Brazilian high schools, and others scenarios that seldom appear in the literature. Moreover, we introduce a new way to model the STP, which is very flexible and allows for an efficient evaluation of the objective function. The developed GRASP algorithms use a local search procedure that interleaves two types of movements and a path-relinking strategy that is enhanced with a local search procedure. Our approach was tested using real data from different Brazilian educational institutions. Despite some restrictions that are specific to the Brazilian high schools, we believe that the ideas introduced in this work can inspire similar approaches for solving the STP in other educational institutions.

The article is organized as follow. Section 2 describes the school timetabling problem treated in this work. Section 3 introduces the problem representation model, the neighborhoods and the objective function. Sections 4 and 5 describe, respectively, the GRASP and GRASP with path-relinking algorithms. Section 6 shows the computation results for real instances and Section 7 offers some conclusions.

2 Problem Definition

The STP in question consists of a set of m classes, n teachers, s subjects and p weekly periods of time. The latter are prefixed for each class. Classes are disjoint sets of students, who attend the same lectures. Each topic of a given class is associated with only one teacher, who is previously determined. Periods are distributed in d weekly days and in h daily periods that occur during the same shifts for each class, i.e., $p = d \times h$. But importantly, d and h can vary from class to class and daily periods can start and finish at different times. As a consequence, overlaps can occur between daily periods of different classes.

As an example, Figure 1 shows the daily periods of three different classes. Each period has a duration of 50 minutes. Classes $C1$ and $C3$ have five periods and class $C2$ has six periods. The gray rectangles represent the break interval of each class. As can be seen, the first period of class $C1$ overlaps with the first period of class $C2$ for 30 minutes (7:30 – 8:00). Another case of overlap occurs between the fourth periods of classes $C2$ and $C3$ and it lasts for 20 minutes (10:30 – 10:50).

The objective of the problem is to determine a sequence of meetings between teachers and classes according to the predefined class periods, while optimizing a set of operational constraints. The following constraints are considered:

(a) Each class's curriculum cannot be violated;

(b) A class cannot have a lecture with more than one teacher at the same time;

(c) A class cannot have more than two lectures of the same subject in the same day. If the subject is just taught twice a week, they cannot be scheduled in the same day;

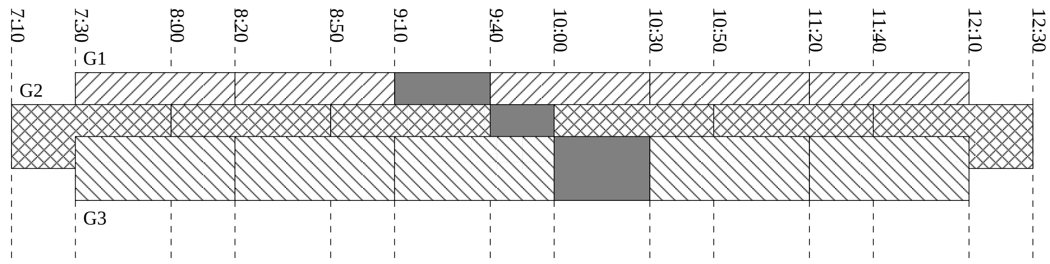


Figure 1: Three diferent timetables of Class

(d) Each teacher must fulfill his/her weekly number of lectures;

(e) A teacher cannot be scheduled to a period that he/she is not available;

(f) A teacher's waiting time between two lectures must be less than a class period;

(g) Lectures defined as simultaneous must be scheduled at the same period;

(h) A teacher cannot teach more than one class at the same time, except if the lectures must be taught simultaneously;

(i) A teacher is entitled to mark a number of days where he/she would prefer not to work;

(j) A teacher is entitled to mark a number of periods where he/she would prefer not to work;

(k) Certain lectures like art, music and foreign language, should be equitably distributed during the week;

(l) Two lectures of the same subject that occur in the same day should be scheduled in two consecutives periods.

The constraints can be divided into two sets. The first is called the set of hard constraints, and it includes constraints (a) – (h). In a feasible timetable, all hard constraint must be satisfied. The second set is the set of soft constraints, and it includes constraints (i) – (l). The number of soft constraints violated should be minimized. Hence, a practicable timetable must not violate any hard constraints and should strive to satisfy all soft ones.

Despite the fact that the constraints (f) and (g) are essential to many instances of the STP, they are seldom treated in the literature. The reason is that they significantly increase the complexity of the problem. We believe that constraints (i), (j) and (k) are more typical of Brazilian high schools, since we have not found similar constraints considered in the literature.

3 A New Way To Model the STP

This section describes the data structures, two different neighborhoods and the objective function that are used in the algorithms. The data structures illustrate a new way to model the problem and allow an efficiently evaluation of the objective function. The neighborhoods are generated through two different types of movement, including a double movement that attempts to correct the imperfections of the first type.

3.1 Solution Representation

Classes may have periods starting and finishing in distinct times. Moreover, the number of lectures per week may vary between classes. For instance, some of the real instances considered came from the Stella Maris school, located at the city of Santos, in Brazil. In that school, timetables are:

Timetable-1 has 5 lectures per day, starting at 7:30, 8:20, 9:40, 10:30 and 11:20;

Timetable-2 has 5 lectures per day, starting at 7:30, 8:20, 9:10, 10:30 and 11:20;

Timetable-3 has 6 lectures per day, starting at 7:10, 8:00, 8:50, 10:00, 10:50 and 11:40.

The difficult to solve and model this problem stems from the fact that some teachers can work in classes that follow different timetables. Therefore, two kinds of constraints are raised:

(1) A teacher cannot teach more than one class whose periods start and finish at the same time, except if the lectures must be taught simultaneously;

(2) A teacher cannot teach more than one class whose periods overlap.

Constraints of type (2) arise because the periods of two classes may start and finish at distinct times. Although this type of constraint has not appeared in the literature, it is typicall in Brazilian high schools. In order to treat this particular type of constraint we developed a flexible model, which allows for distinct class timetables, while still efficiently evaluating the objective function. We call this new model the Multi-Timetable model.

This model divides the classes into K sets according to the characteristics of each class timetable. Each set has a particular class timetable, which is represented by a matrix T^k, for $k \in \{1, 2, 3 \ldots K\}$. Line i in T^k represents the weekly schedule for class i. Each element $t_{ij} \in T^K$ indicates the subject scheduled to class i in period j. Each matrix T^k has dimension $m_k \times p_k$, where m_k and p_k are the number of classes and periods in k, respectively. The matrices T^k represent a solution to the problem. The Multi-Timetable model automatically satisfies constraints (a), (b) and (d).

3.2 Neighborhoods

In the algorithms that are discussed later, two types of movements are considered, each one giving rise to different neighborhoods. The first one, called a simple movement, consists of a mere change of two distinct elements of a same line in a matrix T^k, provided that they represent distinct subjects. It is represented by the quadruple $<k, c, p_1, p_2>$, where k identifies the class timetable, c identifies the class and p_1, p_2 represent two distinct periods.

The set of solutions generated by the execution of a simple movement over a solution S defines a neighborhood to the problem, here called $N_1(S)$. This neighborhood contains many neighbors of worse quality than S, because this simple movement just check if the subjects are distinct; consequently, all kinds of violations that are not automatically guaranteed by the Multi-Timetable model can arise.

To attenuate this limitation, we defined a second type of movement, called a double movement, that is composed by two distinct simple movements where the second component is an attempt to correct the possible violations produced by the first. It is represented by two quadruples $< <k_1, c_1, p_{11}, p_{21}>, <k_2, c_2, p_{12}, p_{22}> >$, where k_i identifies the class timetable, c_i identifies the class and p_{1i}, p_{2i} represent periods, for $i \in \{1, 2\}$. However, it is expensive to identify violated constraints produced by a simple movement and to generate another movement that tries to improve over the previous. Instead we try to generate a special second movement only if the first component violates one of the hard constraints (c), (e), (g) and (h). If some violation of these constraints is produced by the first component, the second component will be generated in the following way:

- Assume that two lectures must be simultaneous and this constraint is satisfied in a solution S. If the first component of a double movement is applied to S and produces a solution that violates this constraint, the second component will produce another solution, where the two lectures are simultaneous again;
- If the first component schedules a teacher for a period that he/she is not available, the second component will reschedule this teacher for a period that he/she is available;
- If the first component schedules a teacher for a period that he/she is already scheduled to teach other class, the second component will reschedule this teacher for a period that he/she is available and where he/she is not teaching any class;
- If the first component schedules a subject a for a day where two or more subjects a are already scheduled, the second component will reschedule one of the subjects a for a different day.

The set of solutions generated by the execution of a double movement over a solution S defines a neighborhood to the problem, here called $N_2(S)$. This neighborhood has its limitations too; for instance, when more than one violation is detected after the first component of the double movement is applied, the second component will fix just one of them, according to the following order: (g) $\rightarrow$ (h) $\rightarrow$ (e) $\rightarrow$ (c). Furthermore, the second component can produce a solution of worse quality than that produced by the first one. In practice, however, the double movement is very efficient to find good local minimum solutions, as will be seen.

3.3 Objective Function

A solution S is evaluated according to the following objective function that should be minimized:

$$f(S) = f_{hard}(S, \alpha) + f_{soft}(S, \beta)$$

The terms f_{hard} and f_{soft} measure, respectively, the hard and soft penalties associated with constraints violations of the respective kind in S. The parameters α and β are vectors of positive weights used to penalize each unsatisfied constraint. These weights are chosen to satisfy the condition: $\alpha >> \beta$. Obviously, solution S is feasible if and only if $f_{hard}(S, \alpha) = 0$.

It is possible to evaluate the objective function $f(S')$, for a solution $S' \in N_1(S)$, in a very fast way, because when a simple movement is applied to a solution S, it just produces a small change in the respective timetable T^k.

Consider a simple movement γ, defined as $<k, c, p_1, p_2>$. When this movement is applied to a solution S, it generates another solution S' that is similar to S. In fact, γ only affects the constraints connected to the class c [†] and to the teachers [‡] who teach class c at periods p_1 and p_2. So, we created auxiliary structures that store the sum of some penalties related to each class and to each teacher. More specifically, to each timetable T^k it is associated a matrix X^k of natural values. Each element $x_{ij} \in X^k$ is the sum of the penalties associated with constraints (c), (k) and (l) that are violated by class i at day j. We also use a matrix Y of natural values, where each element $y_{ij} \in Y$ is the sum of the penalties associated with constraints (f) and (h) violated by teacher i at day j. For violations of the constraint (i), we use a vector Z of natural values, where each element $z_i \in Z$ identifies the days that teacher i does not teach any class. We use these auxiliary structures to store the sum of penalties based on days, because when a simple movement is applied to a solution S, what really happens is a swap between two lectures that are scheduled in the same day or in different days. Thus, it is only necessary to recalculate the penalties for all periods in those days affected by the movement. It is not necessary to store the penalties associated with constraints (g), (e) and (j) in the matrices X^k and Y because to recalculate them we just need to check periods p_1 and p_2.

For a solution $S'' \in N_2(S)$, we can also evaluate the objective function $f(S'')$ in a fast way, treating each component of a double movement as described above.

4 A GRASP Strategy for the STP

The GRASP [Res01, RR02] algorithm we propose, called GRASP-STP, comprises two phases: construction and local search. In the construction phase, we use a partially greedy procedure to build a solution (see Section 4.1). In the local search phase, the built solution is refined by means of a particular hill climbing procedure (see Section 4.2). These phases are repeated by a given number of iterations or until a solution of quality greater or equal than a given threshold has been found. The pseudo-code of the GRASP-STP algorithm for minimizing a objective function is presented in Algorithm 1.

[†] These constraints are (c), (g), (k) and (l).

[‡] These constraints are (e), (f), (h), (i) and (j).

Algorithm 1 GRASP-STP

```
Require: $maxitr > 0$ and $threshold \geq 0$
 1: $f(S^*) \leftarrow \infty$
 2: $itr \leftarrow 1$
 3: repeat
 4:    $S^0 \leftarrow GRASP\text{-}Construction()$
 5:    $S \leftarrow HillClimbing(S^0)$
 6:    if $f(S) < f(S^*)$ then
 7:       $S^* \leftarrow S$
 8:    end if
 9:    $itr \leftarrow itr + 1$
10: until $itr > maxitr$ or $f(S^*) \leq threshold$
11: return $S^*$
```

4.1 The Construction Procedure

We adapted an idea of [SMO03] to the Multi-Timetable model, constructing an initial solution by means of a partially greedy procedure, as described in the sequel.

At first, we sort the more critical periods in decreasing order. We consider a period p_1 to be more critical than a period p_2 if the numbers of lectures that can be scheduled in p_1 is smaller than the number that can be scheduled in p_2. Next, we order the lectures based on the values of a greedy function g, generating a list $\mathcal{C}$. The function g is defined as:

$$g(l) = \nu g_1(t) + \xi g_2(t) + \rho g_3(t) + \zeta g_4(l),$$

where ν, ξ, ρ and ζ are weights, and where t is the teacher who teaches lecture l. The function $g_1(t)$ returns the number of unavailable periods that t has, $g_2(t)$ returns the number of lectures taught by t, $g_3(t)$ returns the number of different class timetables in which t is scheduled to teach and $g_4(l)$ returns one if the lecture l must be simultaneous with another lecture or zero otherwise.

Next, a restricted candidate list (RCL) of length determined by a parameter $\alpha \in [0, 1]$ is built. An element (lecture) belongs to the RCL, if the following condition is met:

$$\underline{l} := min\{g(l) | l \in \mathcal{C}\};$$

$$\overline{l} := max\{g(l) | l \in \mathcal{C}\};$$

$$RCL := \{l \in \mathcal{C} | g(l) \geq \overline{l} + \alpha(\underline{l} - \overline{l})\}.$$

Therefore, for $\alpha = 0$, the RCL contains only the lectures that are most critical to be scheduled, according to the function g. When $\alpha = 1$, all lectures that are not yet scheduled will belong to the RCL. Hence, for $\alpha = 0$, $0 < \alpha < 1$ and $\alpha = 1$, the procedure is, respectively, greedy, partially greedy and random.

Next, a lecture is randomly selected from the RCL and, using the list of critical periods, it is scheduled. As a result, more constraints are satisfied. Every time a lecture is scheduled, the critical periods list is updated as well as the list $\mathcal{C}$. A pseudo-code for the constructive procedure is presented in Algorithm 2.

Algorithm 2 Construction Procedure

```
Require: $0 \leq \alpha \leq 1$
 1: generate the list of critical Periods $\mathcal{P}$
 2: generate the list of lectures $\mathcal{C}$
 3: $S \leftarrow \emptyset$
 4: while $\mathcal{C} \neq \emptyset$ do
 5:    $\underline{l} := min\{g(l) | l \in \mathcal{C}\}$
 6:    $\overline{l} := max\{g(l) | l \in \mathcal{C}\}$
 7:    $RCL := \{l \in \mathcal{C} | g(l) \geq \overline{l} + \alpha(\underline{l} - \overline{l})\}$
 8:    choose a random lecture $l \in RCL$
 9:    choose the more critical period $p \in \mathcal{P}$ where more restrictions are satisfied
10:    schedule $l$ at $p$
11:    update $\mathcal{P}$
12:    update $\mathcal{C}$
13: end while
14: return $S$
```

4.2 The Local Search Procedure

The proposed hill climbing [SG01] procedure starts from a solution generated by the constructive procedure and, in a mixed way, explores the neighborhoods $N_1(S)$ and $N_2(S)$ of the current solution S using, respectively, simple and double movements (see Section 3.2).

Let $(S \oplus \delta)$ be the solution generated after execution of movement δ, which can be a simple or a double movement, over solution S. The local search algorithm works in the following way. It generates a random double movement θ, whose first component is a simple movement γ and executes γ and θ over the current solution S, producing two new solutions $S' \leftarrow (S \oplus \gamma)$ and $S'' \leftarrow (S \oplus \theta)$. The algorithm selects S' if the condition $f(S') < f(S'') \leq f(S)$ is satisfied, otherwise it selects S'' if the condition $f(S'') \leq f(S') \leq f(S)$ is satisfied. The algorithm stops when a solution of quality greater or equal than a given threshold has been found, or if a solution of quality better than S has not been found after a given number of iterations. The hill climbing pseudo-code is presented in Algorithm 3.

The main advantage of using simple and double movements in a mixed way is that the time spent to find a local minimum is usually smaller, compared to the time needed when these movements are used in independent strategies. Another important characteristic of Algorithm 3 is the use of double movements, which enhance the traditional hill climbing procedure with a soft strategy to escape from a local minimum. That is, the first component can move to a solution of worse quality than the current solution S, but the second one in turn, can move to a solution at least as good as S.

Algorithm 3 Hill Climbing

```
Require: $maxitr > 0$, $threshold \geq 0$ and $S^0$ - initial solution
 1: $S \leftarrow S^0$
 2: $itr \leftarrow 1$
 3: repeat
 4:     Generate a random movement $\theta$
 5:     if $f(S \oplus \gamma) < f(S \oplus \theta)$ and $f(S \oplus \gamma) \leq f(S)$ then
 6:         $itr \leftarrow 1$
 7:         $S \leftarrow S \oplus \gamma$
 8:     else if $f(S \oplus \theta) \leq f(S)$ then
 9:         $itr \leftarrow 1$
10:         $S \leftarrow S \oplus \theta$
11:     else
12:         $itr \leftarrow itr + 1$
13:     end if
14: until $itr > maxitr$ or $f(S) \leq threshold$
15: return $S$
```

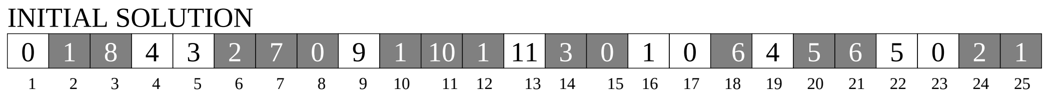
5 A New Path-Relinking Strategy for the STP

Path-relinking is an enhancement to the basic GRASP procedure, usually leading to better solutions. It was originally proposed as an intensification strategy to be used with tabu search [Glo96]. The use of path-relinking within a GRASP procedure was proposed by Laguna and Marti [LM99]. It was followed by several extensions and improvements [ARPT05, ABR03]. The general idea of path-relinking is to explore new solutions along the trajectory that connects two elite solutions. The basic difference among its variations is the way that the trajectory is explored.

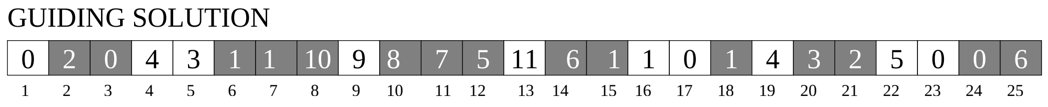
In this section, we propose a path-relinking strategy for the STP. Our approach explores trajectories connecting two elite solutions called, respectively, initial and guiding solutions. This is done by introducing attributes of the guinding solution in the initial one. At each step, we choose a period p of a class c, where the subject scheduled in the initial and guiding solutions are distinct. Then we apply a simple movement that reschedules the subject in the initial solution. Figure 2 shows the subjects scheduled in a same class in an initial and guiding solutions. The gray rectangles are the periods where subjects scheduled are distinct in both solutions. For instance, to make the second period in the initial solution have the same subject that is scheduled in that period in the guiding solution, we should apply to the initial solution one of the following simple movements: $<k, c, 2, 6>$ or $<k, c, 2, 24>$, where k identifies the class timetable and c the class that belongs to k.

Based on this idea, we define a neighborhood $N_{PR}(S_{initial}, S_{guiding})$, where each solution $S \in N_{PR}(S_{initial}, S_{guiding})$ is more similar to $S_{guiding}$ than $S_{initial}$ is.

The conventional path-relinking movements for the STP would consist in following the trajectory that chooses the best solution $S \in N_{PR}(S_{initial}, S_{guiding})$ at each step. However, this approach proved to be poor for STP, because, in practice, its gains are small and are


INITIAL SOLUTION

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	1	8	4	3	2	7	0	9	1	10	1	11	3	0	1	0	6	4	5	6	5	0	2	1


GUIDING SOLUTION

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	2	0	4	3	1	1	10	9	8	7	5	11	6	1	1	0	1	4	3	2	5	0	0	6

Figure 2: The subjects scheduled to the same class in two different solutions

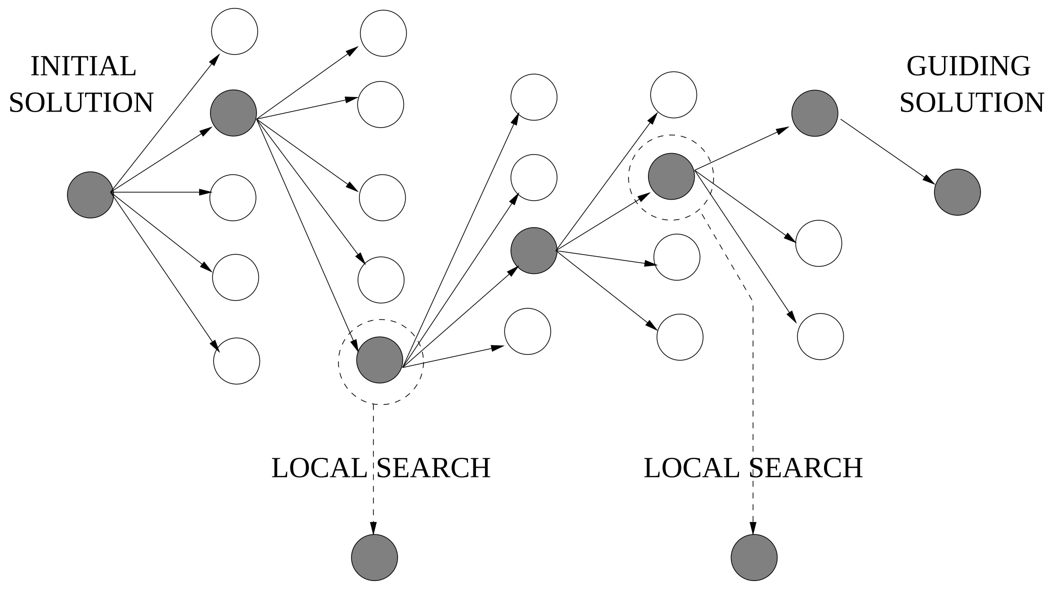


Figure 3: Path-Relinking enhanced with a local search strategy

computational expensive. We improved the conventional path-relinking with a local search procedure. This new strategy is illustrated in Figure 3, where each circle represents a solution to the STP and the set of gray circles defines a trajectory that connects the initial and guiding solutions. At each step, a solution $S \in N_{PR}(S_{initial}, S_{guiding})$ is randomly chosen. A local search procedure is used in determined points in the trajectory, where the condition $f(S) \leq$ *ls-threshold* is satisfied. By doing so, a better solution can be sought both along the explored trajectory and by the local search procedure that is selectively applied during the path-relinking algorithm. This algorithm is described as Algorithm 4. We apply the local search procedure every time the initial solution becomes 10% more similar to the guiding one. We chose this approach because the search procedure is too expensive to be applied at each step in the trajectory.

To enhance the GRASP-STP with the path-relinking strategy, we use a pool of elite solutions. A solution S is an elite solution, if $f(S) \leq$ *pool-threshold*. The initial GRASP iterations are used to fill the pool. In the remaining iterations, a bidirectional path-relinking

Algorithm 4 Path-Relinking

```
Require: $S_{initial}$, $S_{guiding}$ $ls\text{-}threshold > 0$
1: $S_{PR} \leftarrow S_{initial}$
2: $d \leftarrow$ number of distinct periods between $S_{initial}$ and $S_{guiding}$
3: $d \leftarrow d \div 10$
4: $itr \leftarrow 0$
5: while $S_{initial} \neq S_{guiding}$ do
6:     Choose a random solution $S \in N_{PR}(S_{initial}, S_{guiding})$
7:     $S_{initial} \leftarrow S$
8:     if $f(S_{initial}) < f(S_{PR})$ then
9:         $S_{PR} \leftarrow S$
10:    end if
11:    $itr \leftarrow itr + 1$
12:    if $itr = d$ and $f(S_{initial}) \leq ls\text{-}threshold$ then
13:        $d \leftarrow 0$
14:        $S \leftarrow HillClimbing(S_{initial})$
15:        if $f(S) < f(S_{PR})$ then
16:            $S_{PR} \leftarrow S$
17:        end if
18:    else if $itr = d$ then
19:        $d \leftarrow 0$
20:    end if
21: end while
22: return $S_{PR}$
```

is applied if the solutions found by the GRASP are sufficiently good. Every time the path-relinking is performed, the pool is updated, according the following policy:

- If the solution S found by the path-relinking algorithm is better than all solutions in the pool; the worst solution in the pool is replaced by S.
- If the solution S found by the path-relinking algorithm is better than the worst solution in the pool; S replaces the solution of the pool that is more similar to it.

The algorithm GRASP-STP with path-relinking (GRASP+PR-STP) is presented as Algorithm 5. Path-relinking is applied between the current GRASP solution S and a solution S_{pool} randomly chosen from the pool. Path-relinking is applied in two directions: $S \rightarrow S_{pool}$ and $S_{pool} \rightarrow S$.

6 Computational Results

This section reports on results of computational experiments done with the algorithms GRASP-STP and the GRASP+PR-STP proposed, respectively, in Section 4 and in Section 5.

Algorithm 5 GRASP+PR-STP

```
Require: $maxitr > 0$, $max\text{-}pool > 0$, $pool\text{-}threshold > 0$ , $pr\text{-}threshold > 0$,
         $ls\text{-}threshold > 0$ and $GRASP\text{-}threshold \geq 0$
1: $f(S^*) \leftarrow \infty$
2: $pool \leftarrow \emptyset$
3: $itr \leftarrow 1$
4: repeat
5:   $S^0 \leftarrow GRASP\text{-}construction()$
6:   $S \leftarrow HillClimbing(S^0)$
7:   if $|pool| < max\text{-}pool$ and $f(S) < pool\text{-}threshold$ then
8:     $pool \cup S$
9:   else if $|pool| = max\text{-}pool$ and $f(S) < pr\text{-}threshold$ then
10:    choose a random elite solution $S_{pool} \in pool$
11:    $S_{pr1} \leftarrow$ Path-Relinking($S$, $S_{pool}$, $ls\text{-}threshold$)
12:    update $pool$
13:    $S_{pr2} \leftarrow$ Path-Relinking($S_{pool}$, $S$, $ls\text{-}threshold$)
14:    update $pool$
15:    if $f(S_{pr1}) < f(S_{pr2})$ then
16:      $S \leftarrow S_{pr1}$
17:    else
18:      $S \leftarrow S_{pr2}$
19:    end if
20:  end if
21:  if $f(S) < f(S^*)$ then
22:    $S^* \leftarrow S$
23:  end if
24:  $itr \leftarrow itr + 1$
25: until $itr > maxitr$ or $f(S^*) \leq GRASP\text{-}threshold$
26: return $S^*$
```

The experiments were carried out on an Intel Pentium 4 2.40GHz CPU, with 512MB of memory. The algorithms were coded in C++ and were compiled with GCC-4.0.0, using flag -O3.

6.1 Test Problems

Three instances from real Brazilian high schools data were used for the experiments. The dimensions of the problems are summarized in Table 1. The instance stp-1 is a problem of medium size, having 14 classes divided into three different class timetables. The number of teachers who teach classes that belong to more than one timetable is ten and the number of lectures that must be taught simultaneously is six. The instances stp-2 and stp-3 are problems of larger sizes. Each of them has 20 classes divided into two and three different class timetables, respectively. The instance stp-2 has seven and stp-3 has 21 teachers, who teach classes in different timetables. The number of lectures that must be simultaneous, in both instances, is 14. Although the dimensions and the constraints of the problems can vary from one school to another, the instances used on these tests cover the most common cases that are found in Brazil.

Problem	N. of Class Timetables	N. of Classes	N. of Teachers	N. of Lectures
stp-1	3	14	30	385
stp-2	2	20	38	560
stp-3	3	20	38	578

Table 1: Intances from three real Brazilian high schools

6.2 The Experiments

The goal of the experiments was to observe the behavior of the proposed algorithms and to compare their effectiveness. On the two implementations tested in this work, the restricted candidate list parameter α was fixed at 0.6. The parameter *maxitr* of the hill climbing procedure was fixed at $150,000$, if the procedure is activated in the GRASP algorithm itself or at $50,000$, if it is activated during path-relinking. For the GRASP with path-relinking, we used a pool of size $|\mathcal{P}|$ of 5, set *pool-threshold* and *pr-threshold* at 250 and used $10,000$ for the *ls-threshold* parameter.

To study the behavior and the effect of path-relinking with the basic GRASP, we compared the results of both GRASP and GRASP with path-relinking procedures on all three instances. For each instance, we executed 200 independent runs of the two algorithms and each execution was interrupted when a solution with cost lesser than or equal to 30 was obtained. The target value of 30, represents, for these three instances, a solution that satisfies all hard constraints and almost all soft ones.

Figures 4, 5 and 6 show the empirical and theoretical distribution of the solution time for GRASP and GRASP with path-relinking for instances stp-1, stp-2 and stp-3, respectively. To plot these charts, we used the same methodology found in [ARR00], which shows that these distributions fit a two-parameter exponential.

To plot the empirical distribution, first we sorted the time of each independent run of the algorithms in increasing order. Then, we associated with the i-th time t_i a probability $p_i = (i - \frac{1}{2})/200$. We tehn ploted the points $z_i = (ti, pi)$, for $i = 1, \ldots, 200$. Recalling that the cumulative distribution function for the two-parameter exponential is given by

$$F(t) = 1 - e^{-\frac{(t-\mu)}{\lambda}},$$

where λ is the mean and standard deviation of the distribution data and μ is the shift of the distribution with respect to the ordinate axis. For each p_i, $i = 1, \ldots, 200$, we can associate a p_i-quantile $Qt(p_i)$ of the theoretical distribution. For each p_i-quantile we have that

$$F(Qt(pi)) = p_i.$$

Therefore, $Qt(p_i) = F^{-1}(p_i)$. Hence, for the two-parameter exponential distribution, we have that

$$Qt(p_i) = -\lambda ln(1 - p_i) + \mu.$$

To avoid distortions that can be caused by the extreme outliers, we estimate the λ parameter using the upper quartile q_u and lower quartile q_l, that are, respectively, $Q(\frac{1}{4})$ and $Q(\frac{3}{4})$. As a result, we have that

$$\lambda = \frac{z_u - z_l}{q_u - q_l}.$$

Time	stp-1		stp-2		stp-3	
(seconds)	GRASP	GRASP+PR	GRASP	GRASP+PR	GRASP	GRASP+PR
40	0.37	0.45	0.37	0.44	0.36	0.46
80	0.59	0.77	0.60	0.76	0.57	0.82
120	0.73	0.91	0.74	0.89	0.71	0.94
160	0.82	0.96	0.83	0.95	0.81	0.98
200	0.88	0.98	0.89	0.98	0.87	0.99
240	0.92	0.99	0.93	0.99	0.91	0.99

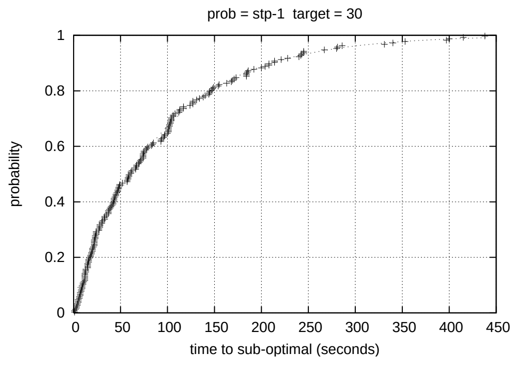
Table 2: Probability estimates of findig a solution at least as good as the target solution. Instances are stp-1, stp2 and stp-3, with target value 30.

The plots illustrated in Figure 7 show a comparison between GRASP-STP and GRASP-+PR-STP. We can observe that GRASP+PR-STP is faster than GRASP-STP when seeking a solution at least as good as the target solution in all tested instances. Table 2 shows the estimated probabilities of finding the target solution as a function of time, for all instances. For example, for a computational time of at most 80 seconds, the estimated probability of finding a solution at least as good as the target solution for instance stp-1 is 59% for the GRASP-STP procedure and 77% for the GRASP+PR-STP algorithm. For a computational time of 160 seconds, this estimated probability for instance stp-2 is 83% for GRASP-STP and 95% for GRASP+PR-STP. On instance stp-3, for a time of 200 seconds, this probability is 87% for GRASP-STP and 99% for GRASP+PR-STP.

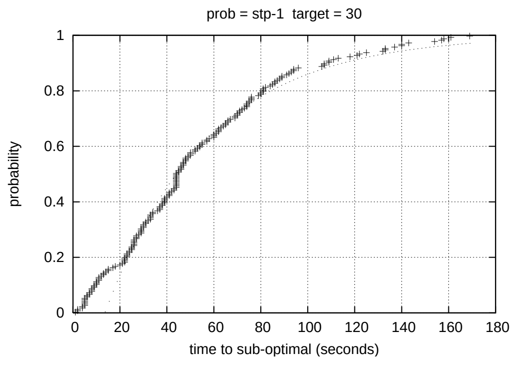
7 Conclusions

This work treats a school timetabling problem with a large and complex set of constraints, including some specific to Brazilian educational institutions and others that are seldom treated in the literature.

We developed a flexible model for this problem, called the Multi-Timetabling model, which allows an efficient evaluation of the objective function. We used two heuristics: a basic GRASP and a GRASP with path-relinking. Both algorithms use a local search procedure that seeks for better solutions through two different neighborhoods and the path-relinking

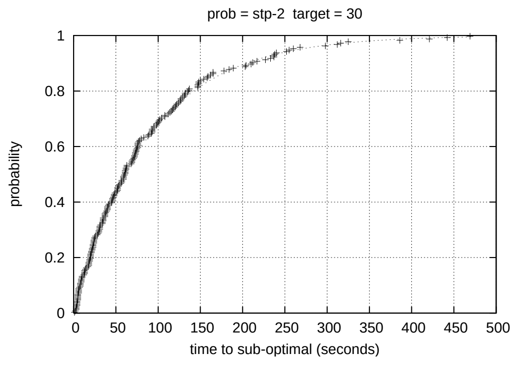


(a)

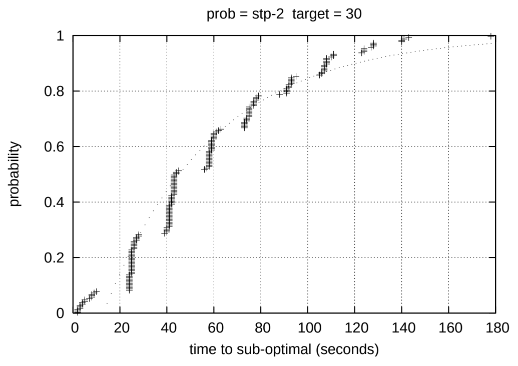


(b)

Figure 4: Distributions for GRASP and GRASP with path-relinking: instance stp-1

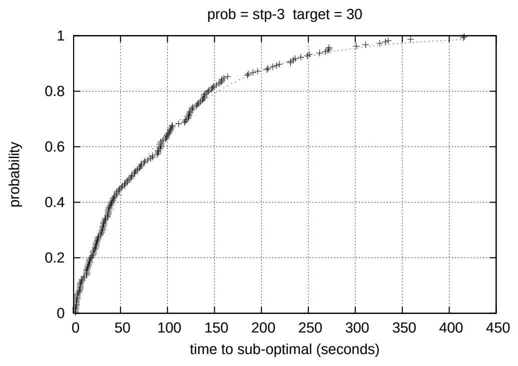


(a)

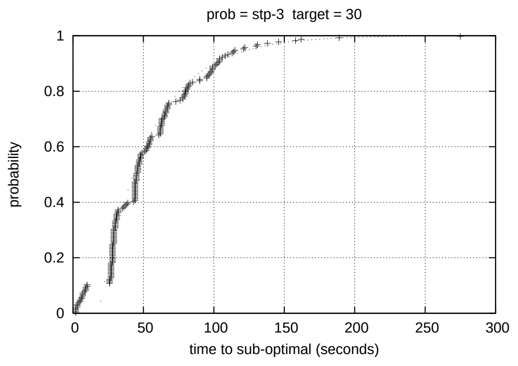


(b)

Figure 5: Distributions for GRASP and GRASP with path-relinking: instance stp-2



(a)



(b)

Figure 6: Distributions for GRASP and GRASP with path-relinking: instance stp-3

strategy is enhanced with a local search procedure that is applied in a selective way. The algorithms were tested using instances from different Brazilian high schools and they proved

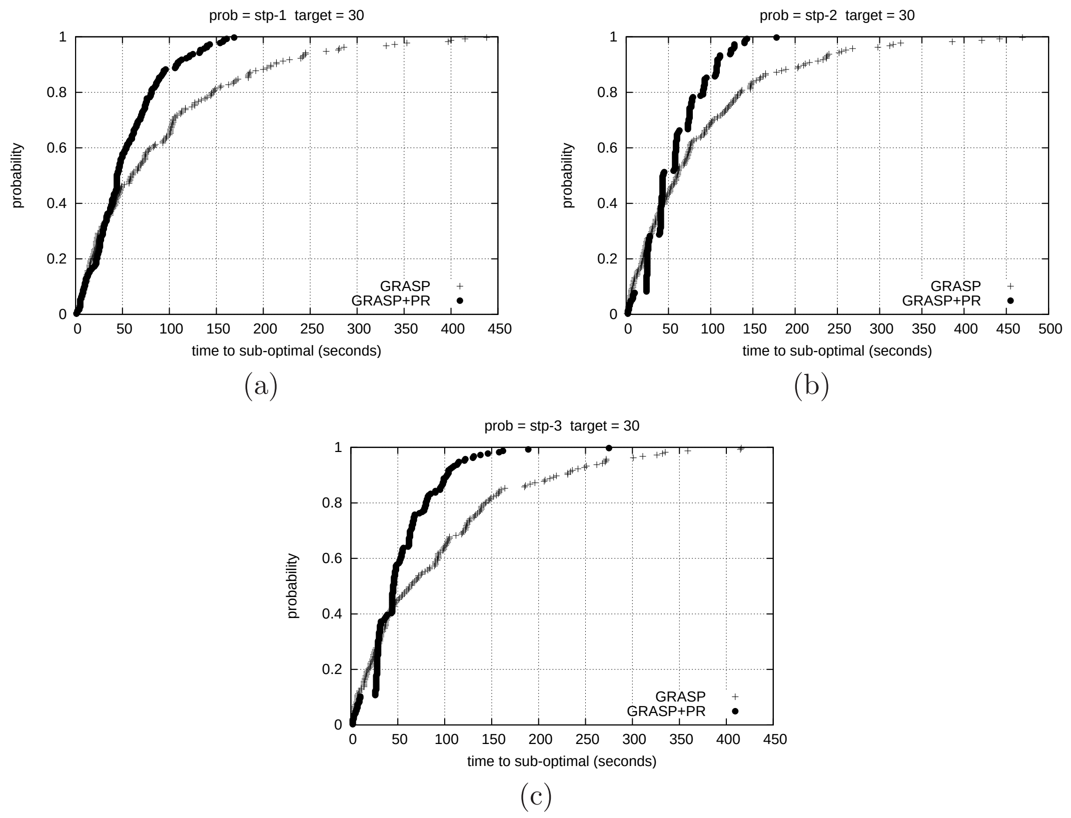


Figure 7: Exponential distribution for GRASP and GRASP with path-relinking

to be suitable to solve the problem in an efficient and fast way.

One point of the algorithms that may be improved is the GRASP construction phase. A possible idea is to explore memory aspects, using the knowledge obtained in the previous iterations to guide the construction procedure. Other local search heuristics like tabu search and simulated annealing can be easily implemented using the Multi-Timetabling model and they may also be alternatives to treat the STP.

References

[ABR03] R. M. Aiex, S. Binato, and M. G. C. Resende. Parallel grasp with path-relinking for job shop scheduling. *Parallel Comput.*, 29(4):393–430, 2003.

[AKD99] D. Abramson, M. Krishnamoorthy, and H. Dang. Simulated annealing cooling schedules for the school timetabling problem. *Asia-Pacific Journal of Operational Research*, 16:1–22, 199.

[ARPT05] R. M. Aiex, M. G. C. Resende, P. M. Pardalos, and G. Toraldo. Grasp with path relinking for three-index assignment. *INFORMS J. on Computing*, 17(2):224–247, 2005.

[ARR00] R. M. Aiex, M. G. C. Resende, and C. C. Ribeiro. Probability distribution of solution time in GRASP: An experimental investigation. Technical Report 00.7.2, Florham Park, NJ, 2000.

[EIS76] S. Evene, A. Itai, and A. Shamir. On the complexity ot timetable and multicommodity flow problems. *SIAM Jornaul of Computing*, 5(4):207–218, 1976.

[Fan94] H. Fang. *Genetic Algorithms in Timetabling and Scheduling.* PhD thesis, University of Edinburgh, Scotland, 1994.

[Glo96] F. Glover. Tabu search and adaptive memory programing – advances, applications and challenges. In *Interfaces in Computer Science and Operations Research*, pages 1–75. Kluwer, 1996.

[Got63] C. C. Gotlieb. The construction of class-teacher timetables. In Cicely M. Popplewell, editor, *IFIP Congress 62*, pages 73–77, Munich, 1963. North-Holland.

[LM99] M. Laguna and R. Marti. Grasp and path relinking for 2-layer straight line crossing minimization. *INFORMS J. on Computing*, 11(1):44–52, 1999.

[NT74] G. A. Neufeld and J. Tartar. Graph coloring conditions for the existence of solutions to the timetable problem. *Commun. ACM*, 17(8):450–453, 1974.

[OdW82] R. Ostermann and D. de Werra. Some experiments with a timetabling system. *OR Spectrum*, 3(4):199–204, 1982.

[Res01] M.G.C. Resende. Greedy randomized adaptive search procedures (GRASP). In *Encyclopedia of Optimization*, volume 2, pages 373–382. Kluwer Academic Publishers, 2001.

[RR02] M.G.C. Resende and C.C. Ribeiro. Greedy randomized adaptive search procedures. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer Academic Publishers, 2002.

[Sch96] A. Schaerf. Tabu search techniques for large high-school timetabling problems. Technical Report CS-R9611, CWI, Amsterdam, NL, 1996. Available at `http://www.cwi.nl/ftp/CWIreports/AP`.

[Sch99] A. Schaerf. A survey of automated timetabling. *Artif. Intell. Rev.*, 13(2):87–127, 1999.

[SG01] A. Schaerf and L. D. Gaspero. Local search techniques for educational timetabling problems (invited paper). In L. Lenart, L. Zadnik Stirn, and S. Drobne, editors, *Proc. of the 6th International Symposium on Operations Research in Slovenia (SOR-01)*, pages 13–23, 2001.

[SMO03] M.J.F. Souza, N. Maculan, and L.S. Ochi. A GRASP-tabu search algorithm for school timetabling problems. In M.G.C. Resende and J.P. de Sousa, editors, *Metaheuristics: Computer decision-making*, pages 659–672. Kluwer Academic Publishers, 2003.

[SS80] G. Schmidt and T. Strohlein. Timetable construction - an annotated bibliography. *Comput. J.*, 23(4):307–316, 1980.

[Tri84] A. Tripathy. School timetabling – a case in large binary integer linear programming. *Management Science*, 30(12):1473–1489, 1984.