

A Tabu Search Heuristic with Efficient Diversification Strategies for the Class/Teacher Timetabling Problem

Haroldo G. Santos¹, Luiz S. Ochi¹, and Marcone J.F. Souza²

¹ Instituto de Computação, Universidade Federal Fluminense, Niterói, Brazil
{hsantos,satoru}@ic.uff.br

² Departamento de Computação, Universidade Federal de Ouro Preto, Ouro Preto, Brazil
marcone@iceb.ufop.br

Abstract. The Class/Teacher Timetabling Problem (CTTP) deals with the weekly scheduling of encounters between teachers and classes of an educational institution. Since CTTP is a NP-hard problem for nearly all of its variants, the use of heuristic methods for its resolution is justified. This paper presents an efficient Tabu Search (TS) heuristic with two different memory based diversification strategies for CTTP. Results obtained through an application of the method for a set of real world problems show that it produces better solutions than a previously proposed TS found in the literature and faster times are observed in the production of good quality solutions.

1 Introduction

The Class/Teacher Timetabling Problem (CTTP) embraces the scheduling of sequential encounters between teachers and students so as to insure that requirements and constraints are satisfied. Typically, the manual solution of this problem extends for various days or weeks and normally produces unsatisfactory results due to the fact that lesson periods could be scheduled which are inconsistent with pedagogical needs or could even serve as impediments for certain teachers or students. CTTP is considered a NP-hard problem [7] for nearly all of its variants, justifying the usage of heuristic methods for its resolution. In this manner, various heuristic and metaheuristic approaches have been applied with success in the solution of this problem, such as: Tabu Search (TS) [14,5,12], Genetic Algorithms [15] and Simulated Annealing (SA) [2].

The application of TS to the CTTP is specially interesting, since this method is, as local search methods generally are, very well suited for the interactive building of timetables, a much recognized quality in timetable building systems. Furthermore, TS based algorithms offer robust solution methods for timetabling problems [6], often presenting the best known solutions, when compared to other metaheuristics [4,13]. The diversification strategy is an important aspect in the design of a TS algorithm. Since the use of a tabu list is not enough to prevent

the search process from becoming trapped in certain regions of the search space, other mechanisms have been proposed. In particular, for the CTTP, two main approaches have been used: adaptive relaxation [12,5] and random restart [14]. In adaptive relaxation the costs involved in the objective function are dynamically changed to guide the search process to newly, unvisited, regions of the search space. In random restart a new solution is generated and no previous information is utilized.

This work employs a TS algorithm that uses informed diversification strategies, which take into account the history of the search process to guide the selection of diversification movements. Although it uses only standard TS components, it provides better results than more complex previous proposals [14].

The article is organized as follows: section 2 presents related works; section 3 introduces the problem to be treated; section 4 presents the proposed algorithm; section 5 describes the computational experiments and their results; and finally, section 6 formulates conclusions and future research proposals.

2 Related Works

Although the CTTP is a classical combinatorial optimization problem, no widely accepted model is used in the literature. The reason is that the characteristics of the problem are highly dependent on the educational system of the country and the type of institution involved. As such, although the basic search problem is the same, variations are introduced in different works (mainly in the evaluation of timetables) [4,5,12,14]. Described afterwards, the problem considered in this paper derives from [14] and considers the timetabling problem encountered in typical Brazilian public high schools. In [14], a GRASP-Tabu Search (GTS-II) metaheuristic was developed to tackle this problem. The GTS-II method incorporates a specialized improvement procedure named “Intraclases-Interclases”, which uses a shortest-path graph algorithm. At first, the procedure is activated aiming to attain the feasibility of the constructed solution, after which, it then aims to improve the feasible solution. The movements made in the “Intraclases-Interclases” also remain with the tabu status for a given number of iterations. Diversification is implemented through the generation of new solutions, in the GRASP constructive phase. In [13] three different metaheuristics that incorporate the “Intraclases-Interclases” were proposed: Simulated Annealing, Micro-canonical Optimization (MO) and Tabu Search. The TS proposal significantly outperformed both SA and MO.

3 The Problem Considered

The problem considered deals with the scheduling of encounters with teachers and classes over a weekly period. The schedule is made up of d days of the week with h daily periods, defining the set P , with $p = d \times h$ distinct periods. There is a set T with t teachers which teach to a set C of c classes, which are disjoint sets of students with the same curriculum. The allocation of teachers to classes is

previously fixed and the workload is informed in a matrix of requirements $R_{t \times c}$, where r_{ij} indicates the number of lessons that teacher i shall teach for class j . Classes are available at any period, and must have their time schedules, of length p , completely filled out, while each teacher i indicates his/her set of available periods A_i . Also, teachers may request a number of double lessons per class. These lessons are lessons which must be allocated in two consecutive periods on the same day. This way a solution to the CTPP problem must satisfy the following constraints:

1. no class or teacher can be allocated for two lessons in the same period;
2. teachers can only be allocated respecting their availabilities;
3. each teacher must fulfill his/her weekly number of lessons;
4. for pedagogical reasons no class can have more than two lesson periods with the same teacher per day.

Also, there are the following desirable features that a timetable should present:

1. the time schedule for each teacher should encompass the least possible number of days;
2. double lessons requests must be satisfied whenever possible;
3. “gaps” in the time schedule of teachers should be avoided, that is: periods of no activity between two lesson periods.

3.1 Solution Representation

A timetable is represented as a matrix $Q_{t \times p}$, in such a way that each row represents the complete weekly timetable for a given teacher. As such, the value $q_{ik} \in \{0, 1, \dots, c\}$, indicates the class for which the teacher i is teaching during period k ($q_{ik} \in \{1, \dots, c\}$), or if the teacher is available for allocation ($q_{ik} = 0$). The advantage of this representation is that it eliminates the possibility for the occurrence of conflicts for teachers. The occurrence of conflicts in classes happens when in a given period k more than one teacher is allocated to that class. Allocations are only allowed in periods with teacher availability. A partial sample of a timetable with 5 teachers can be found in Figure 1, with value “X” indicating unavailabilities of teachers.

Teacher \ Period	1	2	3	4	5	... $d \times h$
1	1	0	0	2	2	...
2	0	X	X	0	1	...
3	X	X	1	0	3	...
4	0	1	0	1	0	...
5	0	0	2	3	X	...

Fig. 1. Fragment of generated timetable

3.2 Objective Function

In order to treat CTTTP as an optimization problem, it is necessary to define an objective function that determines the degree of infeasibility and dissatisfaction of requirements; that is, pretends to generate feasible solutions with a minimal number of unsatisfied requisites. Thus, a timetable Q is evaluated with the following objective function, which should be minimized:

$$f(Q) = \omega \times f_1(Q) + \delta \times f_2(Q) + \rho \times f_3(Q) \quad (1)$$

where f_1 counts, for each period k , the number of times that more than one teacher teaches the same class in period k and the number of times that a class has no activity in k . The f_2 portion measures the number of allocations that disregard the daily limits of lessons (constraint 4). As such, a timetable can only be considered feasible if $f_1(Q) = f_2(Q) = 0$. The importance of the costs involved defines a hierarchy so that: $\omega > \delta \gg \rho$. The f_3 component in the objective function measures the dissatisfaction of personal requests from teachers, namely: double lessons, non-existence of “gaps” and timetable compactness, as follows:

$$f_3(Q) = \sum_{i=1}^t \alpha_i \times b_i + \beta_i \times v_i + \gamma_i \times c_i \quad (2)$$

where α_i , β_i , and γ_i are weights that reflect, respectively, the relative importance of the number of “gaps” b_i , the number of week days v_i each teacher is allocated for teaching, 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 for teacher i .

4 Tabu Search for the Class/Teacher Timetabling Problem

Tabu Search (TS) is an iterative method for solving combinatorial optimization problems. It explicitly makes use of memory structures to guide a hill-descending heuristic to continue exploration without being confused by the absence of improvement movements. This technique was independently proposed by Glover [8] and Hansen [10]. For a detailed description of TS, the reader is referred to [9]. This section presents a brief explanation of TS principles. They are followed by specifications of the customized TS implementation proposed in this paper.

Starting from an initial solution x , the method systematically explores the neighborhood $\mathcal{N}(x)$ and selects the best admissible movement m , so that the application of m in the current solution x (denoted by $x \oplus m$) produces the new current solution $x' \in \mathcal{N}(x)$. Movements that deteriorate the cost function are also permitted. Thus, to try to avoid cycling, a mechanism called short term memory is employed. The objective of short term memory is to try to forbid movements toward already visited solutions, which is usually achieved by the prohibition of the last performed movements. These movements are stored in

a tabu list and remain forbidden (with tabu status), for a given number of iterations, called *tabu tenure*. Since this strategy can be too restrictive, so as not to disregard high quality solutions, movements with tabu status can be accepted if the new solution produced satisfies an *aspiration criterion*. Also, intensification and diversification procedures can be used. These procedures, respectively, aim to deeply investigate promising regions of the search space and to ensure that no region of the search space remains neglected. Following is a description of the constructive algorithm and the customized TS implementation proposed in this paper.

4.1 Constructive Algorithm

The constructive algorithm basically consists of a greedy randomized constructive procedure [11]. Although in other works the option for a randomized construction is to provide diversification, through multiple re-initializations, in our implementation the only purpose is to have control of the randomization degree of initial solution. The construction procedure (Figure 2) is somewhat similar to the human way of building timetables. To build a solution, step-by-step, the principle of allocating first the *most urgent* lessons in the *most appropriate* periods is used. In this case, the urgency degree θ_{ij} of allocating a lesson from teacher i for class j is computed considering the available periods V_i from teacher i , the available periods W_j from class j and the number of unscheduled lessons ζ_{ij} of teacher i for class j , as follows: $\theta_{ij} = \frac{\zeta_{ij}}{|V_i \cap W_j| + 1}$. The algorithm then builds a restricted candidate list (*RCL*) with ordered pairs (i, j) with highest urgency degrees, such that $RCL = \{(i, j) \mid \theta_{ij} \geq \bar{\theta} - (\bar{\theta} - \underline{\theta}) \times \alpha\}$, where $\bar{\theta} = \max\{\theta_{ij} \mid i \in T, j \in C\}$ and $\underline{\theta} = \min\{\theta_{ij} \mid i \in T, j \in C\}$. At each iteration, one lesson from teacher i and class j , so that $(i, j) \in RCL$, is randomly selected for allocation. The α parameter allows tuning the randomization degree of the algorithm, varying from the pure greedy lesson selection ($\alpha = 0$) to a completely random ($\alpha = 1$) selection of teacher and class for allocation.

The selection of the period in which the selected lesson will be allocated is done in free periods of teachers, trying to prevent clashes in classes timetables (this constraint is violated whenever $W_j \cap V_i = \emptyset$). To provide another level of diversity in the initial solution, the selection of period for allocation is also probabilistic, in a way that periods with low availability of teachers will have an exponentially bigger probability of being chosen [3].

At each iteration, the number of unscheduled lessons, availabilities of teachers and classes and urgency degrees are recomputed. The process continues till no more unscheduled lessons are found (i.e., $\zeta_{ij} = 0, i \in T, j \in C$).

4.2 Tabu Search Components

The TS procedure (Figure 3) starts from the initial timetable Q provided by the constructive algorithm and, at each iteration, fully explores the neighborhood $\mathcal{N}(Q)$ to select the next movement m . The movement definition used here is the

```

procedure GenerateTimetable( $\alpha, R, A, P$ )
1:  $\zeta_{ij} \leftarrow r_{ij}$  ( $i \in T, j \in C$ );
2:  $V_i \leftarrow A_i$  ( $i \in T$ );  $W_j = P$  ( $j \in C$ );
3: repeat
4:  $\theta_{ij} = \frac{\zeta_{ij}}{|V_i \cap W_j| + 1}$ ;
5:  $RCL = \{(i, j) \mid \theta_{ij} \geq \bar{\theta} - (\bar{\theta} - \underline{\theta}) \times \alpha\}$ ;
6: Randomly select  $(d, e)$ , such that  $(d, e) \in RCL$ ;
7:  $F \leftarrow V_d \cap W_e$ 
8: if  $F = \emptyset$  then
9:    $F \leftarrow V_d$ ;
10: end if
11: Associate probabilities to periods and randomly select  $f \in F$ ;
12:  $Q_{df} \leftarrow e$ ;
13: until  $\exists \zeta_{ij} > 0$  ( $i \in T, j \in C$ );
14: return  $Q$ ;
end GenerateTimetable.

```

Fig. 2. Pseudo-code for **GenerateTimetable**

same as in [12], and involves the swapping of two values in the timetable of a teacher $i \in T$, which can be defined as the triple $\langle i, p_1, p_2 \rangle$, such that $q_{ip_1} \neq q_{ip_2}$, $p_1 < p_2$ and $p_1, p_2 \in \{1, \dots, p\}$. Clearly, any timetable can be reached through a sequence of these movements that is, at most, the number of lessons in the requirements matrix. Once a movement m is selected, it will be kept in the tabu list during the next $tabuTenure(m)$ iterations. In order to hinder the occurrence of cycling, $tabuTenure(m)$ is not a fixed value, but is randomly selected from values close to a central value (c_{tenure} input parameter). The allowable deviation from this value is defined by the φ input parameter ($\varphi \in [0, 1]$), such that it will determine the range of possible values for tabu tenure (line 19). Insertions and removals in tabu list can be made at every iteration (line 20). The aspiration criterion defined is that the movement will lose its tabu status if its application produces the best solution found so far (line 11).

Since short term memory is not enough to prevent the search process from becoming entrenched in certain regions of the search space, some diversification strategy is necessary. In the proposed method, long term memory is used to guide the diversification procedure. The motivation to employ a memory guided diversification procedure instead of random re-start is twofold: firstly, information loss incurred from random re-start is avoided and secondly, the use of memory to guide the diversification process, hopefully, diminishes the risk of revisiting the same region of the search space.

Two types of long term memory are proposed. The first type regards the storage of transition measures, counting the frequency of movements involving each teacher and class. The second type regards the storage of residency measures counting the number of times in which each lesson has occupied a given period. Every time a movement is done, long term memory information is updated (line

```

procedure ImproveTimetable( $Q, divActivation, iterationsDiv, c_{tenure}, \varphi$ )
1:  $Q^* = Q; TabuList = \emptyset; noImprovementIterations = 0; iteration = 0;$ 
2: initializeLongTermMemory();
3: repeat
4:    $\Delta = \infty; iteration ++;$ 
5:   for all movement  $m$  such that  $(Q \oplus m) \in \mathcal{N}(Q)$  do
6:      $penalty = 0;$ 
7:     if ( $noImprovementIterations \bmod divActivation < iterationsDiv$ )
8:       and ( $iteration \geq divActivation$ ) then
9:          $penalty = computePenalty(m);$ 
10:        end if
11:         $\Delta' = f(Q \oplus m) - f(Q);$ 
12:        if ( $(\Delta' + penalty < \Delta)$  and ( $m \notin TabuList$ )) or ( $f(Q \oplus m) < f(Q^*)$ ) then
13:           $bestMov = m;$ 
14:           $\Delta = \Delta';$ 
15:          if ( $f(Q \oplus m) \geq f(Q^*)$ ) then  $\Delta = \Delta + penalty;$ 
16:          end if
17:        end for
18:        updateLongTermMemory( $bestMov, Q$ );
19:         $Q = Q \oplus bestMov;$ 
20:         $tabuTenure(bestMov) = random(\lfloor c_{tenure} - \varphi \times c_{tenure} \rfloor, \lceil c_{tenure} + \varphi \times c_{tenure} \rceil);$ 
21:        updateTabuList( $bestMov, iteration$ );
22:        if ( $f(Q) < f(Q^*)$ ) then
23:           $Q^* = Q; noImprovementIterations = 0;$ 
24:          initializeLongTermMemory();
25:        else
26:           $noImprovementIterations ++;$ 
27:        end if
28:      until termination criterion reached;
29:      return  $Q^*;$ 
end ImproveTimetable.

```

Fig. 3. Pseudo-code for tabu search algorithm to the class/teacher timetabling problem

17), and every time the best solution is updated, long term memory is cleared (line 23).

While the diversification strategy is active, long term memory information is used to guide the selection of movements, so that movements in slight modified timetables and/or movements which make unusual allocations are encouraged. This is done through the incorporation of penalties in the evaluation of movements (line 8). In the following paragraphs a description of the proposed long term memories and how they are used to compute penalties in the diversification strategy is presented.

Transition based long term memory: in this type of memory, transition measures are stored in a matrix $Z_{t \times c}$, counting how many movements z_{ij} were

done involving teacher i and class j . Using these values, transition ratios are computed. Let $\bar{z} = \max\{z_{ij} \mid i \in T, j \in C\}$, the transition ratio ϵ_{ij} for teacher i and class j is:

$$\epsilon_{ij} = \frac{z_{ij}}{\bar{z}} \quad (3)$$

Since a movement can involve two lesson periods, or a lesson period and a free period, the penalty used in the diversification strategy $\psi_{ia_1a_2}$ associated with a movement in the timetable of teacher i , in periods p_1 and p_2 with allocations $a_1 = q_{ip_1}$ and $a_2 = q_{ip_2}$, respectively, considering the cost of the current solution $f(Q)$ is:

$$\psi_{ia_1a_2} = \begin{cases} \epsilon_{ia_1} \times f(Q) & \text{if } a_1 \neq 0 \text{ and } a_2 = 0 \\ \epsilon_{ia_2} \times f(Q) & \text{if } a_1 = 0 \text{ and } a_2 \neq 0 \\ (\epsilon_{ia_1} + \epsilon_{ia_2})/2 \times f(Q) & \text{if } a_1 \neq 0 \text{ and } a_2 \neq 0 \end{cases}$$

Residence based long term memory: in this type of memory, residence measures are stored for each lesson, in a $Y_{t \times c \times u \times p}$ matrix ($u = \max\{r_{ij} \mid i \in T, j \in C\}$), where z_{ijmk} expresses how many iterations the m -th lesson of teacher i on class j occupied period k . Although it is a fourth-dimensional matrix, this is a very sparse matrix, in a way that efficient implementations make its use practical for problems considered in this paper. To compute the residence ratio η_{ijmk} of the m -th lesson of teacher i and class j on period k , the maximum value of y_{ijmk} ($i \in T, j \in C, m \in \{1, 2, \dots, u\}, j \in P$) \bar{y} is considered, as follows:

$$\eta_{ijmk} = \frac{y_{ijmk}}{\bar{y}} \quad (4)$$

Thus, the penalty μ_{ijmk} for allocating the m -th lesson of teacher i and class j on period k is:

$$\mu_{ijmk} = \eta_{ijmk} \times f(Q) \quad (5)$$

For movements which involve two allocations in a timetable for a given teacher the penalty will be the average penalty of the involved lessons.

The diversification strategy is applied whenever signals that regional entrenchment may be in action are detected. In this case, the number of non-improvement iterations is evaluated before starting the diversification strategy (line 7). The number of non-improvement iterations necessary to start the diversification process (*divActivation*) and the number of iterations that the process will remain active (*iterationsDiv*) are input parameters. The process is cyclic and restarts whenever a multiple of *divActivation* non-improvement iterations is reached. Movements performed in this phase can be viewed as *influential movements* [9], since these movements try to modify the solution structure in a influential (non-random) manner. The function *computePenalty* (line 8) can use one of the proposed long term memory based penalties. In the following sections

the tabu search implementation with transition based long term memory will be referred as TST, while the implementation with residence based long term memory will be referred as TSR. Another implementation, which maintains both types of long term memory will be referred as TSTR. In TSTR, penalties computed using transition based long term memory and residence based long term memory are summed and used in the diversification strategy.

For comparison purposes, an implementation without any diversification strategy (TS), will also be considered in the next sections.

5 Computational Experiments and Discussion

Experiments were done in the set of instances originated from [14], and the data referred to public Brazilian high schools, with 25 lesson periods per week for each class, in different shifts. In Table 1 some of the characteristics of the instances can be verified, such as dimension and sparseness ratio (sr), which can be computed considering the total number of lessons ($\#lessons$) and the total number of unavailable periods ($\#u$): $sr = \frac{t \times p - (\#lessons + \#u)}{t \times p}$. Lower sparseness values indicate more restrictive problems and, likewise, problems in which it is more difficult to find feasible timetables.

Instance	Teachers	Classes	Total Lessons	Double Lessons	Sparseness Ratio (sr)
1	8	3	75	21	0.43
2	14	6	150	29	0.50
3	16	8	200	4	0.30
4	23	12	300	66	0.18
5	31	13	325	71	0.58
6	30	14	350	63	0.52
7	33	20	500	84	0.39

Table 1. Characteristics of problem instances

Three objectives guided the selection of computational experiments to be included in this work: firstly, to search for the best parameters and modules composition (which diversification strategy gives better results), secondly, verify how the proposed tabu search heuristic compares to the previously proposed GTS-II algorithm, and thirdly, verify how good the provided solutions are, considering its practical application.

The algorithms were coded in C++ and the implementation of GTS-II was the same presented in [14]. The compiler used was GCC 3.2.3 using flag `-O2`. Experiments were performed in micro-computers with AMD Athlon XP 1533 MHz processors, 512 megabytes of RAM, running the Linux operating system.

The weights in the objective function were defined as in [14]: $\omega = 100$, $\delta = 30$, $\rho = 1$, $\alpha_i = 3$, $\beta_i = 3$ and $\gamma_i = 1$, $\forall i = 1, \dots, t$.

TS						TST				
Instance	Central Tabu Tenure				Average	Central Tabu Tenure				Average
	15	20	25	30		15	20	25	30	
1	5.13	2.32	0.78	1.28	2.38	0.35	0.30	0.15	0.10	0.22
2	3.47	2.21	1.15	1.72	2.14	0.00	0.12	0.52	0.38	0.25
3	7.94	6.68	2.48	0.92	4.50	0.46	1.03	0.27	0.00	0.44
4	0.97	0.96	0.82	0.24	0.75	0.36	0.27	0.27	0.31	0.30
5	4.83	6.32	1.96	0.45	3.39	0.22	0.21	0.24	0.37	0.26
6	3.45	2.14	0.94	0.30	1.71	0.19	0.27	0.45	0.45	0.34
7	1.75	1.65	0.54	0.82	1.19	0.00	0.19	0.43	0.55	0.29
Average	3.93	3.18	1.24	0.82	2.29	0.22	0.34	0.33	0.31	0.30

TSR						TSTR				
Instance	Central Tabu Tenure				Average	Central Tabu Tenure				Average
	15	20	25	30		15	20	25	30	
1	0.59	0.15	0.00	0.20	0.23	0.25	0.10	0.15	0.25	0.19
2	0.61	0.32	0.20	0.41	0.39	0.26	0.44	0.61	0.61	0.48
3	0.14	0.60	0.76	1.17	0.66	0.73	0.30	0.53	0.53	0.52
4	0.24	0.21	0.34	0.37	0.29	0.00	0.15	0.19	0.43	0.19
5	0.30	0.00	0.35	0.31	0.24	0.05	0.09	0.56	0.55	0.31
6	0.00	0.39	0.39	0.41	0.30	0.08	0.19	0.53	0.59	0.35
7	0.45	0.63	0.38	0.72	0.55	0.04	0.20	0.36	0.44	0.26
Average	0.33	0.33	0.34	0.51	0.38	0.20	0.21	0.42	0.49	0.33

Table 2. Average distance from best known solutions, for each instance, in different tabu search strategies

Initially, experiments to verify which is the best parameter configuration for the proposed algorithms were done (parameters for GTS-II were the same used in [14]). Average results of 10 independent executions (different random seeds) on each instance for different central tabu tenure values (c_{tenure}) and instances were computed (other parameters remain fixed: $\alpha = 0.1$, $\varphi = 0.1$, $divActivation = 500$ and $iterationsDiv = 10$). Executions had fixed time limits, as proposed in [14], which are for instances $\{1, \dots, 7\}$: $\{90, 280, 380, 870, 1930, 1650, 2650\}$ seconds, respectively. In Table 2 the average distance of the cost of generated solutions from the best know solutions is shown. As can be seen, for TS (without diversification strategy), better results were obtained with the highest c_{tenure} values. Nevertheless, implementations with the proposed diversification strategies obtained better results, with any c_{tenure} value, than the simple TS. While on average TST performed better than TSR, the best results were obtained in the implementation which considers both types of long term memory, using low c_{tenure} values, since TSTR with $c_{tenure} = 15$ generated solutions, on average,

only 0.2% off from best known solutions. From now on, results of the proposed algorithms consider experiments with parameters which produced better average results (i.e.: for c_{tenure} : 30 for TS and 15 for TST, TSR and TSTR).

A different view of the results of the previously described experiment is presented in Table 3. Average solution costs generated by proposed algorithms are compared to average results of GTS-II within the same time limits. Best results are shown in bold.

Instance	GTS-II	TS	TST	TSR	TSTR
1	204.80	205.30	203.40	203.00	203.20
2	350.10	349.20	343.30	344.40	344.20
3	455.70	440.90	438.90	439.50	440.10
4	686.30	670.50	671.30	670.30	668.90
5	796.30	782.70	780.90	779.20	779.60
6	799.10	781.50	780.70	782.20	779.80
7	1,076.20	1,063.80	1,055.20	1,061.90	1,055.60

Table 3. Average results, runs with fixed time limits

Inst.	Constructive Algorithm					TSTR				
	$f_1(Q^*)$	$f_2(Q^*)$	#d (%d)	#g (%g)	cr	$f_1(Q^*)$	$f_2(Q^*)$	#d (%d)	#g (%g)	cr
1	0.0	0.5	15.1 (71.5)	17.2 (22.9)	1.6	0.0	0.0	1.9 (9.0)	4.1 (5.5)	1.2
2	0.0	0.0	24.3 (83.8)	24.8 (16.5)	1.3	0.0	0.0	7.3 (25.2)	1.3 (0.9)	1.0
3	0.3	2.5	2.0 (50.0)	31.2 (15.6)	1.4	0.0	0.0	0.4 (10.0)	5.4 (2.7)	1.1
4	4.3	0.9	35.5 (53.8)	21.0 (7.0)	1.2	0.0	0.0	19.4 (29.4)	3.8 (1.3)	1.0
5	0.0	0.2	54.1 (76.1)	46.4 (14.3)	1.5	0.0	0.0	13.7 (19.3)	3.5 (1.1)	1.1
6	0.2	0.0	53.7 (85.2)	53.4 (15.3)	1.4	0.0	0.0	15.4 (24.4)	8.8 (2.5)	1.0
7	0.5	0.2	69.6 (82.9)	74.1 (14.8)	1.3	0.0	0.0	23.0 (27.4)	10.6 (2.1)	1.0

Table 4. Average costs of objective function components obtained by the constructive algorithm and at the end of the tabu search heuristic TSTR

As can be seen in Table 3, although only minor differences can be observed among implementations that use different penalty functions in the diversification strategy, results show that versions using informed diversification strategies perform significantly better than GTS-II and TS.

In order to evaluate the quality of the solutions obtained by the proposed method, taking into account its practical application, and to verify how significant is the improvement of TSTR over the solution received from the constructive algorithm, Table 4 presents the average costs involved in each objective function component, for the solution provided by the constructive algorithm and for the

improved solution from TSTR. Columns $\#d$ ($\%d$), $\#g$ ($\%g$) and cr are related to the f_3 component of the objective function, in the following way: $\#d$ ($\%d$) indicates the unsatisfied double lessons (and the percentage of unsatisfied double lessons, considering the number of double lesson requests), while $\#g$ ($\%g$) indicates the number of “gaps” in the timetable of teachers (and the percentage considering the total number of lessons) and cr measures the compactness ratio of timetable of teachers. To compute cr , the summation of the actual number of days ad that each teacher must teach some lesson in the school in a given timetable and the lower bound for this value \underline{ad} are used. The \underline{ad} value considers the minimum number of days $md_i = \lceil \frac{\sum_{j=1}^c r_{ij}}{h} \rceil$ that each teacher i must teach some lecture in the school, such that $\underline{ad} = \sum_{i=1}^t md_i$. This way, $cr = ad/\underline{ad}$. Values close to one indicate that the timetable is as compact as it can be. As can be seen in Table 4, the solution provided by the constructive algorithm usually contains some type of infeasibility. These problems were always solved by the TSTR algorithm, in a way that no infeasible timetable was produced. Regarding the preferences of teachers, the timetable compactness, which has the highest weight in the f_3 component of the objective function, it can be seen that in most cases the optimal value was reached ($cr = 1$). Also, small percentage values of “gaps” and unsatisfied double lessons were obtained.

Instance	GTS-II			TSTR		
	25%	50%	75%	25%	50%	75%
1	7.64	9.57	12.15	2.13	3.36	6.39
2	21.39	26.57	34.68	9.03	13.48	19.71
3	28.57	46.84	85.41	16.29	27.66	46.47
4	49.22	92.57	146.50	2.65	3.40	5.45
5	47.79	62.85	102.20	27.63	37.85	54.51
6	35.81	48.00	72.12	25.20	33.97	44.38
7	92.41	150.72	287.48	89.57	118.82	155.72

Table 5. Time (in seconds) for 25%, 50% and 75% of runs achieve the target solution values.

In another set of experiments, the objective was to verify the empirical probability distribution of reaching a given sub-optimal target value (i.e. find a solution with cost at least as good as the target value) in function of time in different instances. The sub-optimal values were chosen in a way that the slowest algorithm could terminate in a reasonable amount of time. In these experiments, TSTR and GTS-II were evaluated and the execution times of 150 independent runs for each instance were computed. The experiment design follows the proposal of [1]. The results of each algorithm were plotted by associating the i -th smallest running time t_i with a probability $p_i = (i - \frac{1}{2})/150$, which generates points $z_i = (t_i, p_i)$, for $i = 1, \dots, 150$. The results show that TSTR achieves high probability values

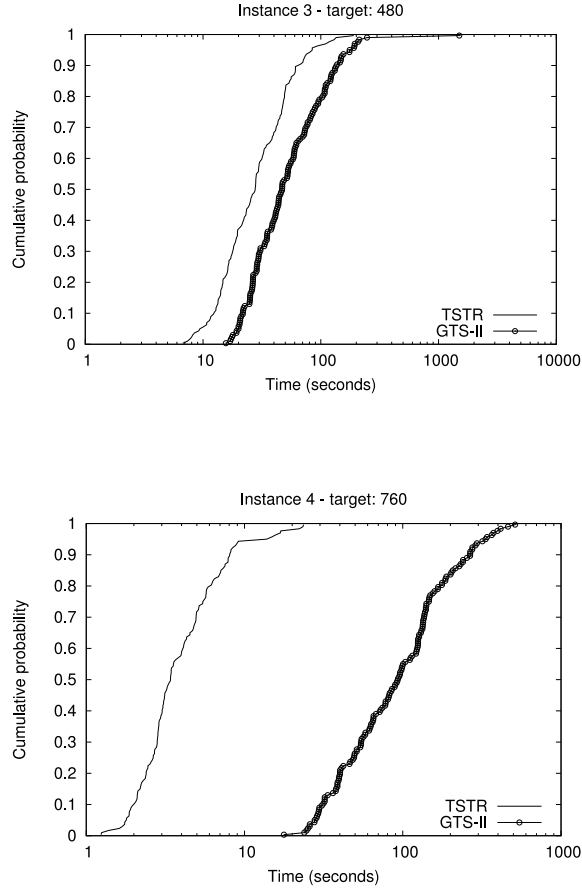


Fig. 4. Empirical probability distribution of finding target value in function of time for instances 3 and 4

($\geq 50\%$) of reaching the target values in significantly smaller times than *GTS-II*, for all instances. Representative results are presented in figures 4 and 5.

This difference is enhanced mainly in instance 4, which presents a very low sparseness ratio. This result may be related to the fact that the “Intraclass-Interclasses” procedure of *GTS-II* works with movements that use free periods, which are hard to find in this instance. Another analysis, taking into account all test instances, shows that at the time when 95% of *TSTR* runs have achieved the target value, in average, only 64% of *GTS-II* runs have achieved the target value. Considering the time when 50% of *TSTR* runs have achieved the target value, only 11%, in average, of *GTS-II* runs have achieved the target value. Table 5 presents the execution times needed by *GTS-II* and *TSTR* to achieve different probabilities of reaching the target values.

6 Concluding Remarks

This paper presented a new tabu search heuristic to solve the class/teacher timetabling problem. Experiments in real world instances showed that the proposed method outperforms significantly a previously developed hybrid tabu search algorithm, and it has the advantage of a simpler design.

Contributions of this paper include the empirical verification that although informed diversification strategies are not commonly employed in tabu search implementations for the class/teacher timetabling problem, its incorporation can significantly improve the method robustness. The proposed method not only produced better solutions for all test instances but also performed faster than a hybrid tabu search approach.

Although in the proposed algorithm long term memory was used to guide diversification procedures, intensification strategies which use this type of information can be formulated, and its application is worthy of receiving further investigation.

Other interesting enhancement to the algorithm could be the combination of the “Intra-classes-Inter-classes” procedure with an informed diversification strategy, which could lead to even better results.

Acknowledgements

This work was partially supported by CAPES and CNPq. The authors would like to thank Olinto C. B. Araújo, from DENSIS-FEE-UNICAMP, Brazil for their valuable comments on the preparation of this paper.

References

1. Aiex, R. M., Resende, M. G. C., Ribeiro, C. C.: Probability distribution of solution time in GRASP: an experimental investigation, *Journal of Heuristics*, **8** (2002), 343–373
2. Abramson, D.: Constructing school timetables using simulated annealing: sequential and parallel algorithms. *Management Science*. **37** (1991) 98–113
3. Bresina, J.L.: Heuristic-biased stochastic sampling. In: *Proceedings of the AAAI-96*. **1** (1996) 271–278
4. Coloni, A., Dorigo, M., Maniezzo, V.: Metaheuristics for High-School Timetabling. *Computational Optimization and Applications*. **9** (1998) 277–298
5. Costa, D.: A Tabu Search algorithm for computing an operational timetable. *European Journal of Operational Research Society*. **76** (1994) 98–110
6. Dowland, K. A.: Off-the-peg or made to measure: timetabling and scheduling with SA and TS. In: *Practice and Theory of Automated Timetabling II*. Springer Lecture Notes in Computer Science, Vol. 1408. Springer-Verlag, New York (1997) 37–52
7. Even, S., Itai, A., Shamir, A.: On the complexity of timetabling and multicommodity flow problems. *SIAM Journal of Computation*. **5** (1976) 691–703

8. Glover, F.: Future paths for integer programming and artificial intelligence. *Computers & Operations Research*. **13** (1986) 533–549
9. Glover, F., Laguna, M.: *Tabu Search*. Kluwer Academic Publishers, Boston Dordrecht London (1997)
10. Hansen, P.: The steepest ascent mildest descent heuristic for combinatorial programming. *Congress on Numerical Methods in Combinatorial Optimization*. Capri (1986)
11. Resende, M.G.C., Ribeiro. C.C.: Greedy randomized adaptive search procedures. *Handbook of Metaheuristics*. Kluwer. (2003) 219–249
12. Schaerf, A.: *Tabu search techniques for large high-school timetabling problems*. Report CS-R9611. Centrum voor Wiskunde en Informatica, Amsterdam (1996)
13. Souza, M.J.F.: *Programação de Horários em Escolas: Uma Aproximação por Metaheurísticas*, D.Sc. Thesis (in Portuguese), Universidade Federal do Rio de Janeiro - Rio de Janeiro (2000)
14. Souza, M.J.F., Ochi, L.S., Maculan, N.: A GRASP-Tabu search algorithm for solving school timetabling problems. In: Resende, M.G.C., Souza, J.P. (eds.): *Metaheuristics: Computer Decision-Making*. Kluwer Academic Publishers, Boston (2003) 659–672
15. Wilke, P, Gröbner, M., Oster, N.: A hybrid genetic algorithm for school timetabling. In: *AI 2002: McKay B. and Slaney J. (eds.): Advances in Artificial Intelligence*. Springer Lecture Notes in Computer Science, Vol. 2557. Springer-Verlag, New York (2002) 455–464

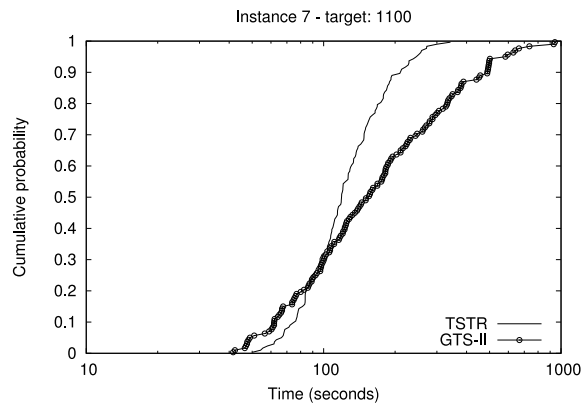
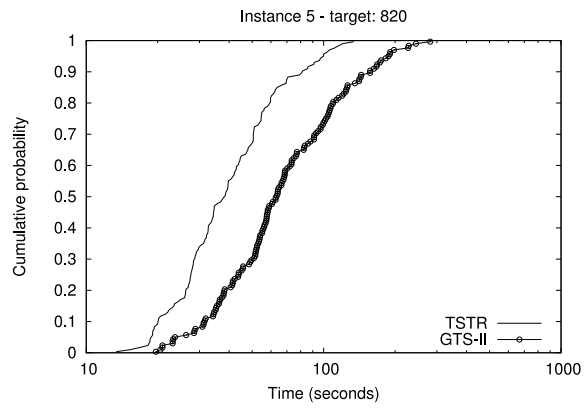


Fig. 5. Empirical probability distribution of finding target value in function of time for instances 5 and 7