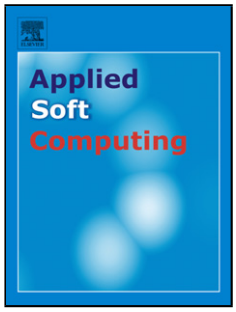


Title: Effective local search algorithms for high school timetabling problems

Author: Landir Saviniec Ademir Aparecido Constantino

PII: S1568-4946(17)30390-3
DOI: http://dx.doi.org/doi:10.1016/j.asoc.2017.06.047
Reference: ASOC 4316

To appear in: *Applied Soft Computing*

Received date: 28-4-2017
Revised date: 23-6-2017
Accepted date: 23-6-2017

Please cite this article as: Landir Saviniec, Ademir Aparecido Constantino, Effective local search algorithms for high school timetabling problems, <*![CDATA[Applied Soft Computing Journal]]*> (2017), http://dx.doi.org/10.1016/j.asoc.2017.06.047

Effective local search algorithms for high school timetabling problems

Landir Saviniec[a,*], Ademir Aparecido Constantino[b]

[a]*Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, Avenida Trabalhador São-carlense 400, 13566-590, São Carlos, São Paulo, Brazil*
[b]*Departamento de Informática, Universidade Estadual de Maringá, Avenida Colombo 5790, 87020-900, Maringá, Paraná, Brazil*

Abstract

This paper addresses the high school timetabling problem. The problem consists in building weekly timetables for meetings between classes and teachers with the goal of minimizing violations of specific requirements. In the last decades, several mixed-integer programs have been proposed and tested for this family of problems. However, medium and large size instances are still not effectively solved by these programs using state-of-the-art solvers and the scientific community has given special attention to the devising of alternative soft computing algorithms. In this paper, we propose a soft computing approach based on Iterated Local Search and Variable Neighborhood Search metaheuristic frameworks. Our algorithms incorporate new neighborhood structures and local search routines to perform an effective search. We validated the proposed algorithms on variants of the problem using seven public instances and a new dataset with 34 real-world instances including large cases. The results demonstrate that the proposed algorithms outperform the state-of-the-art approaches in both cases, finding the best solutions in 38 out of the 41 tested instances.

Keywords: Class-Teacher Timetabling, Iterated Local Search, Variable Neighborhood Search, Minimum Cost Assignment Problem, Conflict Graph.

1. Introduction

Educational timetabling problems consist in building timetables for meetings between teachers (or exams) and students. The scientific literature has branched educational timetabling problems in three main categories: University Course Timetabling Problems [1], Examination Timetabling Problems [2, 3] and High School Timetabling Problems (HSTP) [4, 5, 6]. Each category contains a rich set of specific constraints.

In this paper, we deal with the HSTP. Decision versions of the HSTP can be solved in polynomial time by a min-cost network flow algorithm, provided that there is neither unavailability of teachers nor cases of co-teaching [7]. In addition, if the unavailability of teachers is considered, then the problem becomes NP-Complete [8]. Moreover, if the problem includes

*Corresponding author.
Email addresses: landir.saviniec@gmail.com (Landir Saviniec), ademir@din.uem.br (Ademir Aparecido Constantino)

cases of co-teaching, then it also becomes NP-Complete [9]. Variants originating from practical situations usually extend these decision versions with additional specific requirements, as in the case considered in this paper.

Given the complexity of the HSTP. The problem has been extensively studied over the last decades, several mixed-integer programming (MIP) formulations have been proposed [10, 11, 12, 13]. However, even for specific variants, such as the one studied in Souza et al. [14], practical instances are still not effectively solved by the existing MIP programs using state-of-the-art solvers. As an alternative, the scientific community has given special attention to the devising of soft computing algorithms that compute high-quality timetables with cheap computational times. The interested reader is referred to [15, 16, 17, 18, 19] for a closer look at recent soft computing algorithms to HSTP.

In this paper, we deal with the Brazilian HSTP introduced by Souza et al. [14] and considered later by Santos et al. [20, 10] and Dorneles et al. [16, 12]. The problem consists in building weekly timetables for meetings between classes and teachers with the goal of minimizing violations of specific requirements related to pedagogical practices and teachers' preferences. Souza et al. [14] consider requirements, such as: unavailability of teachers [21], idle periods for teachers (periods during which the teacher is not busy, but is busy in earlier and later periods of the same day) [22], compactness in the weekly schedules of teachers [23] and preferences for double lessons (two lessons in consecutive periods within the same day) [23].

To search high-quality timetables for the considered problem, we propose a soft computing approach based on Iterated Local Search (ILS) and Variable Neighborhood Search (VNS) metaheuristic frameworks. The proposed algorithms employ two neighborhood operators that exploit the structure of the problem to perform an effective local search toward good quality solutions.

In order to evaluate the performance of our approach, we conduct a number of experiments with two variants of the problem using literature instances and new instances. Our study helps researchers, decision makers or programmers to develop applied soft computing algorithms to high school timetabling problems, in order to provide high-quality timetables that meet the requirements of teachers and students.

The main contributions of the paper are: (1) New ILS and VNS based algorithms for high school timetabling problems; (2) Two new generic neighborhood operators that can be easily adapted to similar problems; (3) The hybridization between these neighborhood operators improves hugely the effectiveness of local search procedures; (4) The proposed algorithms found optimal solutions for all instances of a literature dataset [14] and performed better than the solver winner of the Third International Timetabling Competition (ITC2011) in 87.50 % of new proposed instances; (5) We collected 34 real-case instances from thirteen high schools located in different towns in the south of Brazil and made available for next studies[1]. This new dataset has cases that are much larger than the real cases provided by the dataset available for the Brazilian HSTP [14]. With this study, we hope to extend the well-established Brazilian benchmark [14].

[1]The proposed dataset and the program of our soft computing algorithms can be downloaded at http://www.gpea.uem.br/benchmark.html

The paper is organized as follows. Section 2 gives a detailed description of the problem. Section 3 presents an overview of related works. Section 4 details each component of our approach. Section 5 presents the computational experiments and comparisons with previous approaches. The paper ends with Section 6, which summarizes the paper by providing some conclusions and suggestions for future research.

2. The high school timetabling problem

The addressed HSTP [14] is motivated by Brazilian high school rules. In these schools, the daily activities are divided into three independent shifts (morning, afternoon and evening) and a weekly timetable is built for each shift. In each shift, there are a set of classes C and a set of teachers T to teach the classes. The timetable for class-teacher meetings is planned for the set of weekdays D, Monday to Friday. For each day, meetings are assigned to a set of periods H of the same duration.

In this problem, classes are disjoint groups of students having the same subjects and being busy for all periods during the week. The input data (or instance) to construct a timetable in a given shift is represented by the following parameters:

- R: a set of tuples $(c \in C, t \in T, \theta \in \mathbb{N}, \lambda \in \mathbb{N}, \mu \in \mathbb{N})$, called requirements. Each requirement $r \in R$ specifies a subject of a class c that is taught by a teacher t (e.g. mathematics, chemistry, etc). Each requirement also specifies the number of weekly meetings θ (or lessons), a maximum number of daily meetings λ, and a minimum number of weekly double lessons μ that are required for the specified subject. For example, the requirements $(c_1, t_1, \theta_1, \lambda_1, \mu_1)$ and $(c_1, t_1, \theta_2, \lambda_2, \mu_2)$ may specify the subjects of mathematics and chemistry of class c_1, that are taught by teacher t_1. Each one specifying its number of weekly meetings, a maximum number of daily meetings and a minimum number of required double lessons.

- U: a set of tuples $(t \in T, d \in D, h \in H)$ called teachers' unavailable periods, in which there is a tuple $(t, d, h) \in U$ if teacher t is unavailable at period h of day d.

Given these definitions, the problem consists in building a weekly timetable for all meetings, satisfying hard constraints and minimizing soft constraints' violations. The following constraints are considered:

Hard constraints

hc_1: each requirement $r \in R$ must be assigned exactly θ times a week.
hc_2: a class must attend exactly one meeting per period.
hc_3: a teacher must teach at most one lesson per period.
hc_4: teachers must not be scheduled to periods in which they are unavailable.
hc_5: each requirement $r \in R$ must not have more than λ assignments per day.

Soft constraints

sc_1: each requirement $r \in R$ should have at least μ double lessons a week.
sc_2: idle periods in the schedule of teachers should be avoided.
sc_3: the teachers' schedules should be concentrated on a minimum number of days (compactness).

3. Related works

The first studies dedicated to automating high school timetables date from 1960s [24, 25]. Since then, many papers have appeared in the literature, from studies of timetabling complexity, with basic decision versions of the problem [7, 8, 9], to real cases involving a number of different requisites. The most up-to-date review of HSTP and solution techniques can be found in the paper of Pillay [6].

Up to the late 2010s, the research in high school timetabling had been mostly based on independent case studies, addressing no common problems and introducing barriers to the practice of comparing different solution techniques. On the best of our knowledge, the only datasets available were the dataset of Souza et al. [14], addressing problems of Brazilian high schools, and the dataset of Beligiannis et al. [26], addressing cases from Greek high schools.

The Greek HSTP has been addressed in several studies [26, 27, 28, 15, 19]. The state-of-the-art method for the Beligiannis' dataset[2] is a hybrid cat swarm optimization based algorithm proposed by Skoullis et al. [19].

The Brazilian HSTP, that we consider in this paper, has also been the subject of several studies. The problem appeared first in Souza et al. [14]. The authors proposed a hybrid heuristic approach with GRASP and Tabu Search. Other two heuristic approaches for this problem appeared in Santos et al. [20] and Dorneles et al. [16]. Santos et al. [20] proposed a Tabu Search with diversification strategies and Dorneles et al. [16] proposed several versions of Fix-and-Optimize heuristics. The problem has also been addressed by MIP techniques. Santos et al. [10] proposed a Cut and Column Generation algorithm. Dorneles et al. [16] proposed a compact MIP formulation and Dorneles et al. [12] proposed an alternative compact MIP formulation and a column generation. These authors [14, 20, 16, 10, 12] tested their methods in the Souza's dataset[3]. This dataset has seven instances, comprising small and medium sizes, that were proved optimal by comparing column generation lower bounds with integer solutions found in large runs of different heuristic algorithms [10, 16].

In the PATAT conferences of 2008 and 2010, the community of timetabling researchers launched a project[4] to create a format to represent general high school timetabling problems and to create a unified set of benchmark instances. The project resulted in the development of the XHSTT format [29]. This is an XML-based format that established a common set of constraints and objectives taken from different high school timetabling problems found around the world [5]. Most of them, found in papers addressing isolated case studies.

The XHSTT format provided a more general model to represent HSTP problems. After the launch of the XHSTT project, new instances from different countries have been added to the XHSTT repository. At this moment, there are around 25 instances available (not including deprecated ones). The instances of the XHSTT repository were used in the ITC2011 [30], dedicated to high school timetabling problems. The state-of-the-art method for these instances is the solver GOAL [17], which won the competition. An updated version of GOAL is described in Fonseca et al. [18], which employs hybrid algorithms using metaheuristics and matheuristics. The general HSTP specified by the XHSTT format has also been investigated

[2]Available in http://www.deapt.upatras.gr/cso_timetabling/school-timetabling.html
[3]Available in http://labic.ic.uff.br/Instance/
[4]https://www.utwente.nl/ctit/hstt/

by MIP techniques. Kristiansen et al. [11] proposed a first MIP formulation. An alternative improved formulation appeared in Fonseca et al. [13]. The latter is an extension of the formulation proposed by Dorneles et al. [12], which models the HSTP of Brazilian high schools.

As mentioned before, the HSTP has also been the subject of many other case-studies. But this discussion is out of the scope of this paper. For the interested reader, we refer to Pillay [6] for a detailed literature review. In the next section, we concentrate on our soft computing approach.

4. The proposed approach

We propose versions of the ILS [31] and VNS [32] metaheuristics to address the HSTP. The most important idea of our algorithms is the hybridization between two neighborhood operators that exploit the structure of the problem, and how the local search is performed to effectively explore the solution space. Before presenting the algorithms, we define their basic elements: the timetable encoding (Section 4.1), the objective function (Section 4.2), the procedure to construct initial solutions (Section 4.3) and the neighborhood operators (Section 4.4). The proposed algorithms are explained in Section 4.5.

4.1. Timetable encoding

Let P be the set of timeslots $(d, h) \in D \times H$. We represent timetables by a two-dimensional array Z in which the rows represent classes and the columns represent timeslots. Each array's cell Z_{cj} points to a tuple $r \in R$ that is the requirement assigned to class $c \in C$ in timeslot $j \in P$. This assignment means that teacher $r.t$ meets with class c in timeslot j to teach the subject associated with requirement r. An illustration is shown in Figure 1, for a partial timetable where three meetings have been assigned to class c_1. The requirement tuple r_1 has been assigned to class c_1 in timeslots j_1 and j_2, and the tuple r_2 has been assigned to timeslot $j_{|P|}$.

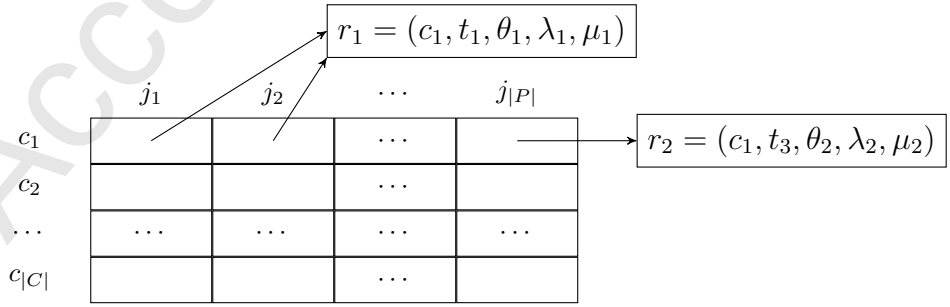


Figure 1: Illustration of the timetable encoding.

This structure allows the definition of a straightforward constructive algorithm to build initial solutions (Section 4.3), that ensure the satisfaction of constraints hc_1 and hc_2. Thus, we do not need to explicitly consider these constraints in the objective function.

4.2. Objective function

Let β_i^{hc} and β_i^{sc} be the number of times that constraint types hc_i $(i = 3, ..., 5)$ and sc_i $(i = 1, ..., 3)$ are violated, and let α_i^{hc} and α_i^{sc} be associated penalty constants to penalize violations. The objective function minimizes the following weighted sum:

$$min \quad f(Z) = \sum_{i=3}^{5} \alpha_i^{hc} \cdot \beta_i^{hc} + \sum_{i=1}^{3} \alpha_i^{sc} \cdot \beta_i^{sc} \tag{1}$$

The first term of (1) measures the feasibility and the second term measures the quality of a timetable Z. The values of α_i^{hc} must be larger than α_i^{sc} to ensure that hard constraints are satisfied.

In our implementation, we used the following penalties: feasibility ($\alpha_3^{hc} = 100.000$, $\alpha_4^{hc} = 100.000$, $\alpha_5^{hc} = 10.000$) and quality ($\alpha_1^{sc} = 1$, $\alpha_2^{sc} = 3$, $\alpha_3^{sc} = 9$). The quality parameters are the same values used by previous approaches. The values of β_i^{hc} and β_i^{sc} are computed as follows:

$\beta_3^{hc} = \sum_{t \in T} \sum_{j=1}^{|P|} (\pi_{tj} - 1)$, $\forall(\pi_{tj} > 1)$, where π_{tj} is the total number of meetings assigned to teacher t in timeslot j.

$\beta_4^{hc} = \sum_{t \in T} \sum_{j=1}^{|P|} \rho_{tj}$, where $\rho_{tj} = 1$ if teacher t is assigned to teach at an unavailable timeslot j, and $\rho_{tj} = 0$ otherwise.

$\beta_5^{hc} = \sum_{r=1}^{|R|} \sum_{d \in D} (\sigma_{rd} - \lambda_r)$, $\forall(\sigma_{rd} > \lambda_r)$, where σ_{rd} is the number of assignments that the r-th requirement has in day d, and λ_r is the maximum number of meetings allowed per day.

$\beta_1^{sc} = \sum_{r=1}^{|R|} (\mu_r - \phi_r)$, $\forall(\mu_r > \phi_r)$, where μ_r is the minimum number of double lessons specified by the r-th requirement, and ϕ_r is the total number effectively scheduled.

$\beta_2^{sc} = \sum_{t \in T} \sum_{d \in D} \eta_{td}$, where η_{td} is the number of idle periods occurring in the schedule of teacher t on day d.

$\beta_3^{sc} = \sum_{t \in T} \chi_t$, where χ_t is the total number of working days scheduled to teacher t.

4.3. Constructive algorithm

Initial solutions are constructed by the randomized heuristic shown in Algorithm 1. This algorithm constructs a random timetable Z with the set of requirements R received as input. This construction ensures that constraints hc_1 and hc_2 are satisfied.

4.4. Neighborhood operators

We propose two neighborhood operators named as *Matching* (MT) and *Torque* (TQ). The first is based on the resolution of Minimum Cost Assignment Problems (MCAPs), and the second is based on connected components of conflicting meetings that are identified by

Algorithm 1 Pseudo-code of the constructive algorithm.

```
CONSTRUCT-SOLUTION(R)
1  Initialize an empty solution Z
2  for each (e ∈ R) do
3      Let p(e) be a pointer to e.
4      c' = e.c
5      numLessons = e.θ
6      while (numLessons > 0) do
7          Let P_c' be the subset of timeslots j ∈ P for which Z_c'j is empty to class c'.
8          Choose a random timeslot j ∈ P_c'.
9          Z_c'j = p(e)
10         numLessons = numLessons − 1
11 return Z
```

conflict graphs. Two meetings are said to conflict with each other if they are scheduled on the same timeslot and they have the same teacher.

The MT operator is inspired by the works of Constantino et al. [33] and Lü et al. [34]. Constantino et al. [33] employed successive assignment problems to heuristically solve the nurse scheduling problem. Lü et al. [34] used assignment problems to allocate rooms after they had assigned the lessons to periods. Our MT operator solves an MCAP to find the best local permutation for a subset of requirements that are assigned to a given class. An illustration is shown in Figure 2, for the subset of requirements $\widehat{R} = \{10, 11, 12, 13\}$ assigned to class c_3 in timeslots $\widehat{P} = \{j_1, j_2, j_3, j_5\}$ of the timetable Z, shown in Figure 2(a). The timetable Z contains four conflicting meetings. Requirements 14 and 19 have the same teacher (teacher number 6) and are assigned to the same timeslot j_1. Requirements 11 and 20 have the same teacher (teacher number 9) and are assigned to the same timeslot j_2. For simplicity, we suppose that only constraint hc_3 is under consideration and $\alpha_3^{hc} = 1$. Therefore, there are two violations of constraint hc_3 and $f(Z) = 2$. By removing requirements $\widehat{R} = \{10, 11, 12, 13\}$ from the timetable Z, we get the partial solution $\widehat{Z}$ shown in Figure 2(b), with objective value $f(\widehat{Z}) = 1$. Checking the possibility of reassigning each requirement of $\widehat{R}$ to each timeslot of $\widehat{P}$, we construct the MCAP's square cost matrix M shown in Figure 2(c). Solving the MCAP, we get the minimum cost assignment solution shown in Figure 2(d), and a new improved neighboring solution Z', shown in Figure 2(e), with objective value $f(Z') = 1$. For the sake of simplicity, in Figure 2 we considered only the constraint type hc_3 to construct the cost matrix M. But in our implementation, all constraints have been considered. To solve the MCAP, we employ the polynomial time algorithm proposed by Carpaneto and Toth [35].

The TQ operator is inspired by the idea of *chain of moves* [34]. The operator builds a graph to identify connected components of requirements that cause conflicting meetings when they are exchanged between two timeslots. Vertices which may cause conflicting meetings are connected by edges. After the connected components are identified, each component forms a chain of one or more two-swap moves that generates a new neighboring solution with a non-increasing number of conflicts. An illustration is shown in Figure 3. Let's consider the timetable Z shown in Figure 3(a). Figure 3(b) shows the conflict graph for requirements assigned to timeslots j_3 and j_4 of the timetable Z in Figure 3(a). In this graph, each node

Classes	d_1				
	j_1	j_2	j_3	j_4	j_5
c_1	1[10]	2[7]	3[15]	4[17]	3[15]
c_2	5[1]	6[6]	7[5]	8[4]	9[7]
c_3	10[2]	11[9]	12[1]	11[9]	13[6]
c_4	14[6]	15[10]	16[12]	17[2]	18[11]
c_5	19[6]	20[9]	20[9]	21[8]	22[10]

(a) A timetable solution Z with conflicting meetings and objective value $f(Z) = 2$. The first number at each cell is a pointer to a requirement tuple (represented here by a non-negative integer value) and the number enclosed by brackets represents the identifier of the associated teacher.

Classes	d_1				
	j_1	j_2	j_3	j_4	j_5
c_1	1[10]	2[7]	3[15]	4[17]	3[15]
c_2	5[1]	6[6]	7[5]	8[4]	9[7]
c_3				11[9]	
c_4	14[6]	15[10]	16[12]	17[2]	18[11]
c_5	19[6]	20[9]	20[9]	21[8]	22[10]

(b) The partial timetable $\widehat{Z}$, with objective value $f(\widehat{Z}) = 1$, after removing the subset of requirements $\widehat{R} = \{10, 11, 12, 13\}$ assigned to class c_3 in timeslots $\widehat{P} = \{j_1, j_2, j_3, j_5\}$.

	j_1	j_2	j_3	j_5
$r = 10$	1	1	1	1
$r = 11$	1	2	2	1
$r = 12$	2	1	1	1
$r = 13$	2	2	1	1

(c) The MCAP's square cost matrix M, in which the rows represent the requirements of $\widehat{R}$ and the columns represent the timeslots of $\widehat{P}$. A cell M_{ij} is the cost of reassigning the i-th requirement of $\widehat{R}$ to the j-th timeslot of $\widehat{P}$.

	j_1	j_2	j_3	j_5
$r = 10$	1	1	1	1
$r = 11$	1	2	2	1
$r = 12$	2	1	1	1
$r = 13$	2	2	1	1

(d) The solution of the MCAP. The assignment of minimum cost is indicated by values shaded with blue color.

Classes	d_1				
	j_1	j_2	j_3	j_4	j_5
c_1	1[10]	2[7]	3[15]	4[17]	3[15]
c_2	5[1]	6[6]	7[5]	8[4]	9[7]
c_3	10[2]	12[1]	13[6]	11[9]	11[9]
c_4	14[6]	15[10]	16[12]	17[2]	18[11]
c_5	19[6]	20[9]	20[9]	21[8]	22[10]

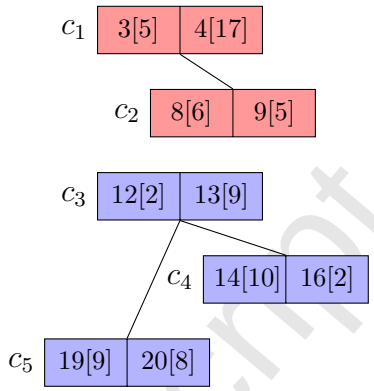
(e) The improved neighboring solution Z', with $f(Z') = 1$, generated from the solution of the MCAP.

Figure 2: Illustration of the MT neighborhood operator.

represents a class and the two requirements assigned to the class in timeslots j_3 and j_4. We connect two vertices if they have the same teacher in opposite timeslots. For example, vertices c_1 and c_2 have teacher 5 (the number enclosed by brackets) assigned to opposite timeslots and they generate conflicting meetings if only one of the vertices has its requirements swapped between the two timeslots. Thus, both vertices must have their requirements swapped between timeslots j_3 and j_4. Figure 3(c) shows a conflict-free neighboring solution Z' of Z, after swapping the requirements that are in the connected component formed by vertices c_1 and c_2 of the graph shown in Figure 3(b). Figure 3(d) shows another possible neighboring solution Z'', after swapping the requirements that are in the connected component formed

Classes	d_1				
	j_1	j_2	j_3	j_4	j_5
c_1	1[2]	2[7]	3[5]	4[17]	5[15]
c_2	6[1]	7[4]	8[6]	9[5]	8[6]
c_3	10[5]	11[10]	12[2]	13[9]	13[9]
c_4	14[10]	15[6]	14[10]	16[2]	17[11]
c_5	18[6]	19[9]	19[9]	20[8]	21[10]

(a) A timetable solution Z.



(b) A conflict graph for requirements assigned to timeslots j_3 and j_4 of the timetable Z shown in Figure 3(a).

Classes	d_1				
	j_1	j_2	j_3	j_4	j_5
c_1	1[2]	2[7]	4[17]	3[5]	5[15]
c_2	6[1]	7[4]	9[5]	8[6]	8[6]
c_3	10[5]	11[10]	12[2]	13[9]	13[9]
c_4	14[10]	15[6]	14[10]	16[2]	17[11]
c_5	18[6]	19[9]	19[9]	20[8]	21[10]

(c) A conflict-free neighboring solution Z' after swapping the requirements that are in the connected component formed by vertices c_1 and c_2 of the graph shown in Figure 3(b).

Classes	d_1				
	j_1	j_2	j_3	j_4	j_5
c_1	1[2]	2[7]	3[5]	4[17]	5[15]
c_2	6[1]	7[4]	8[6]	9[5]	8[6]
c_3	10[5]	11[10]	13[9]	12[2]	13[9]
c_4	14[10]	15[6]	16[2]	14[10]	17[11]
c_5	18[6]	19[9]	20[8]	19[9]	21[10]

(d) A conflict-free neighboring solution Z'' after swapping the requirements that are in the connected component formed by vertices c_3, c_4 and c_5 of the graph shown in Figure 3(b).

Figure 3: Illustration of the TQ neighborhood operator.

by vertices c_3, c_4 and c_5 of the graph shown in Figure 3(b).

These neighborhood operators are formally used by our two local search procedures described in Algorithm 2 and Algorithm 3, presented in the next section.

4.5. ILS and VNS based algorithms for HSTP

We propose three versions of the ILS and two of the VNS metaheuristics. The proposed algorithms use two local search procedures based on the first improvement strategy [32]. The first local search (Algorithm 2) employs the TQ operator to explore the neighborhood of the current solution and the second local search (Algorithm 3) employs the MT operator.

The three ILS based algorithms are as follows:

- ILS-TQ: this version is shown in Algorithm 4. The algorithm employs the local search procedure described in Algorithm 2. The perturbation procedure, line 4 of Algorithm 4, consists in accepting a random neighbor of the current solution. The neighboring solution is generated by choosing a random component of a conflict graph constructed for two randomly chosen timeslots $i, j \in P$, $i \neq j$. This perturbation procedure is used by all of our algorithms.

- ILS-MT-TQ: this version is the Algorithm 4 with the local search phase, in line 5, replaced by a sequence of local search procedures. The algorithm runs the local search

Algorithm 2 Pseudo-code of the local search with the TQ operator.

LocalSearchTQ(Z)

```
1  Let P be the set of timeslots defined in Section 4.1.
2  Let CC be the set of connected components in a graph G of conflicting meetings.
3  do
4      f' = f(Z)
5      for each (i, j ∈ P; i ≠ j) do
6          Construct the graph G for requirements assigned to timeslots i and j of solution Z.
7          Compute the connected components of G and update CC.
8          for each (k ∈ CC) do
9              Swap the requirements at the component k to obtain a neighboring solution Z''.
10             if f(Z'') ≤ f(Z) then
11                 Z = Z''
12 while (f(Z) < f')
13 return Z
```

Algorithm 3 Pseudo-code of the local search with the MT operator.

LocalSearchMT(Z, m)

```
1  Let R̂ be a subset of requirements and P̂ be a subset of timeslots, as explained in Section 4.4.
2  do
3      f' = f(Z)
4      i = m · |C|          // neighborhood size – the number of MCAPs to be explored
5      while (i > 0) do
6          Select a random class c ∈ C.
7          Select a random subset of requirements R̂ assigned to class c in Z and update P̂.
8          Construct the cost matrix M of the MCAP.
9          Solve the MCAP.
10         Update Z based on the MCAP solution.
11         i = i − 1
12 while (f(Z) < f')
13 return Z
```

defined in Algorithm 3 (with parameter $m = 2$), followed by the local search of Algorithm 2. The input of the Algorithm 2 is the local minimum solution returned by Algorithm 3.

- ILS-TQ-MT: this version is similar to the previous algorithm, but we reverse the order of the two local search procedures. The Algorithm 2 is run before Algorithm 3.

The two VNS based algorithms are as follows:

- VNS-MT-TQ: this version is shown in Algorithm 5. The order of neighborhood changes is given by MT neighborhoods, followed by TQ neighborhoods. The algorithm runs the local search defined in Algorithm 3, with $k_{max} - 1$ different sizes for parameter m, followed by the local search of Algorithm 2.

Algorithm 4 Pseudo-code of the ILS-TQ algorithm.

```
ILS-TQ(Z, t_max)
 1  Z* = Z                                          // global best solution
 2  NotImproved = 0
 3  while (CpuTime() < t_max) do
 4      Z = Perturbation(Z, 1)                      // one random move with TQ
 5      Z = LocalSearchTQ(Z)                        // local search
 6      if f(Z) < f(Z*) then
 7          NotImproved = 0
 8      else
 9          NotImproved = NotImproved + 1
10      if f(Z) ≤ f(Z*) then                        // acceptance criterion
11          Z* = Z
12      if NotImproved ≥ 3 then                     // if no improvement after three iterations
13          Z = Z*                                  // return to Z*
14          NotImproved = 0
15  return Z*
```

Algorithm 5 Pseudo-code of the VNS-MT-TQ algorithm.

```
VNS-MT-TQ(Z, t_max, k_max)
 1  Z* = Z
 2  while (CpuTime() < t_max) do
 3      k = 1
 4      do
 5          Z = Perturbation(Z, 1)                  // one random move with TQ
 6          if k ≤ k_max − 1 then
 7              Z = LocalSearchMT(Z, k)
 8          else
 9              Z = LocalSearchTQ(Z)
10          if f(Z) < f(Z*) then
11              k = 1
12          else
13              k = k + 1
14          if f(Z) ≤ f(Z*) then
15              Z* = Z
16          else
17              Z = Z*
18      while (k ≤ k_max)
19  return Z*
```

- VNS-TQ-MT: this version is similar to the previous algorithm, but the local search in Algorithm 2 is run first.

We have set parameter $k_{max} = 7$ to our VNS based algorithms. This value provides a good trade-off between quality of solutions and computational time.

5. Computational experiments

In this section, we conduct two computational experiments in order to evaluate the performance of our soft computing approach. In the first experiment, we compare our algorithms with previous approaches in the dataset of Souza et al. [14]. In the second experiment, we extend the computational study to the new dataset containing larger cases. To simplify our presentation and the discussion of results, we classify the instances by the number of classes, in small ($|C| \leq 10$), medium ($10 < |C| \leq 20$), and large cases ($|C| > 20$).

The proposed algorithms were coded in C++ and compiled with the GNU Compiler Collection 4.9.2. The experiments were performed on a computer with Intel Xeon E7-4860 (2.26 GHz) and 126 GB of RAM, running Debian GNU/Linux 8.8. In all experiments, we carried out 25 independent trials for each instance.

5.1. Experiment with literature instances

This section evaluates the performance of our algorithms in the dataset of Souza et al. [14]. The instances are shown in Table 1. The table presents, for each instance, the number of classes ($|C|$), teachers ($|T|$), days ($|D|$), periods per day ($|H|$), the total number of periods for which the teachers are unavailable ($|U|$), the total number of lessons ($\sum_{e \in R} \theta_e$), the total number of required double lessons ($\sum_{e \in R} \mu_e$) and the classification (Size) in Small (S), Medium (M) or Large (L). In this dataset, only instances 2 to 6 are real cases, the smallest and the largest instances, 1 and 7, were artificially created [10]. The proposed algorithms are compared with previous approaches tested on these instances. For this experiment, the proposed algorithms were set with time limit $t_{max} = 300$ seconds.

Table 1: Features of the instances of Souza et al. [14].

Instance	$\|C\|$	$\|T\|$	$\|D\|$	$\|H\|$	$\|U\|$	$\sum_{e \in R} \theta_e$	$\sum_{e \in R} \mu_e$	Size
1	3	8	5	5	40	75	21	S
2	6	14	5	5	25	150	29	S
3	8	16	5	5	80	200	4	S
4	12	23	5	5	170	300	41	M
5	13	31	5	5	0	325	71	M
6	14	30	5	5	10	350	63	M
7	20	33	5	5	0	500	84	M

The results are shown in Table 2. The values of column LB are lower bounds computed by the cut and column generation (CCG) algorithm proposed by Santos et al. [10]. The asterisk (*) means an optimal solution compared to the lower bounds. Columns ILS-TQ, ILS-MT-TQ, ILS-TQ-MT, VNS-MT-TQ and VNS-TQ-MT show our best results. The column TS shows the results obtained by the Tabu Search of Santos et al. [20]. The authors do not provide the execution time, they only mention that the results were produced by very long running times. The column MIP shows the results of the MIP program of Dorneles et al. [16]. The MIP program was reported to be executed during 10 hours. The column F8 shows the results of the best version of the Fix-and-Optimize heuristics proposed by Dorneles et al. [16], for which we show the computational time enclosed by brackets.

Table 2 shows that our algorithms outperform the previous approaches, as they reach optimal solutions for every instance while the other approaches do not.

Table 2: Best results of our algorithms compared to previous approaches in the instances of Souza et al. [14].

		ILS			VNS				
ID	LB	TQ	MT-TQ	TQ-MT	MT-TQ	TQ-MT	TS	MIP	F8
1	202	*	*	*	*	*	*	*	* [600 s]
2	333	*	*	*	*	*	*	*	* [600 s]
3	423	*	*	*	*	*	*	*	* [21600 s]
4	652	*	*	*	*	*	653	*	* [600 s]
5	762	*	*	*	*	*	766	764	* [600 s]
6	756	*	*	*	*	*	760	765	759 [1800 s]
7	1017	*	*	*	*	*	1029	1028	* [1800 s]

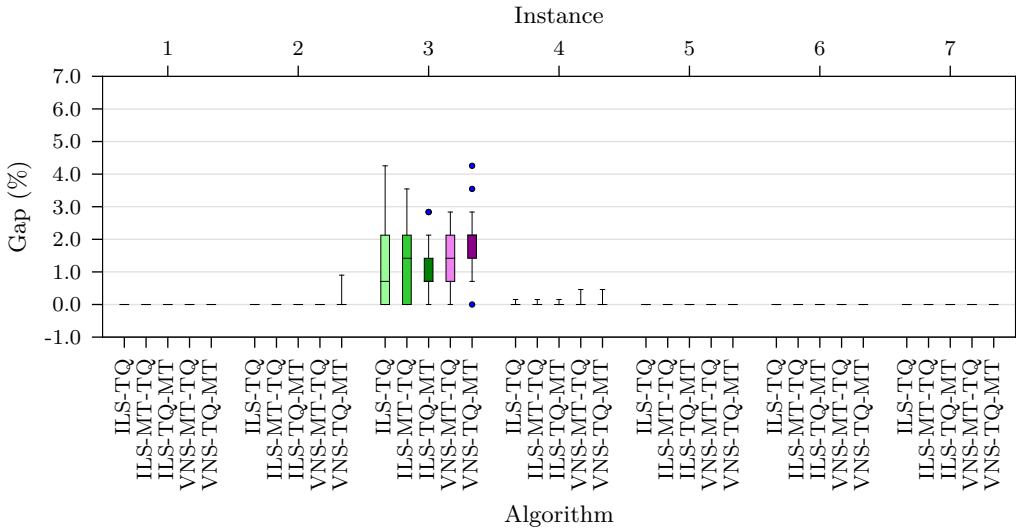


Figure 4: Distribution of solutions' optimality gaps in the instances of Souza et al. [14].

In Figure 4, we show that our algorithms consistently generate optimal and near-optimal solutions for these instances. The figure shows the statistical distribution of solutions' optimality gaps for solutions found in 25 trials. The optimality gap for a given solution Z_i of an instance i is calculated as $Gap = 100 \cdot (f(Z_i) - LB_i)/LB_i$, where LB_i is the best-known lower bound for instance i. The box-plot of Figure 4 shows that the generated solutions are very close to optimal solutions. For instances 1, 2, 4, 5, 6 and 7, the optimal solutions were obtained with high frequency. The algorithms only performed worse in instance 3, in which the gaps are between 0 and 4 %.

5.2. Experiment with new instances

This section evaluates the performance of our algorithms in the new dataset. The dataset comprises 34 real-case instances from years 2008, 2010 and 2011, collected from thirteen high schools in Brazil. The schools are geographically situated in different towns in the State of Paraná. In these cases, many teachers work in more than one school, and usually, they are unavailable in most of the periods for a specific school. Thus, the most complicated situation is the construction of good quality timetables that respect all teachers' unavailable periods. The instances are shown in Table 3. Apart from instances with a large number of

teachers' unavailable periods (column $|U|$), the dataset also presents a diversity of scenarios and instances that are larger than those of Souza et al. [14].

Table 3: Features of the new instances.

Instance	$\|C\|$	$\|T\|$	$\|D\|$	$\|H\|$	$\|U\|$	$\sum_{e\in R}\theta_e$	$\sum_{e\in R}\mu_e$	Size
CM-CEUP-2011-N	3	15	5	5	284	75	36	S
FA-EEF-2011-M	4	12	5	5	160	100	42	S
JNS-CEJXXIII-2011-N	4	15	5	5	12	100	48	S
CM-CEDB-2010-N	5	17	5	5	41	125	60	S
JNS-CEJXXIII-2011-M	5	18	5	5	50	125	60	S
JNS-CEJXXIII-2011-V	5	18	5	5	52	125	60	S
JNS-CEDPII-2011-V	7	21	5	5	101	175	73	S
CM-CECM-2011-N	8	30	5	5	489	200	96	S
JNS-CEDPII-2011-M	8	19	5	5	91	200	85	S
CL-CECL-2011-N-A	9	28	5	5	25	225	107	S
MGA-CEVB-2011-V	9	20	5	5	214	225	97	S
MGA-CEVB-2011-M	10	21	5	5	167	250	108	S
CL-CEASD-2008-V-A	12	27	5	5	108	300	132	M
CL-CEASD-2008-V-B	12	27	5	5	108	300	132	M
MGA-CEDC-2011-V	12	31	5	5	412	300	131	M
CL-CECL-2011-M-A	13	31	5	5	23	325	144	M
CL-CECL-2011-M-B	13	31	5	5	8	325	143	M
CM-CECM-2011-V	13	34	5	5	455	325	142	M
CL-CECL-2011-V-A	14	29	5	5	21	350	164	M
CM-CEUP-2008-V	16	35	5	5	345	400	192	M
CM-CEUP-2011-M	16	38	5	5	498	400	192	M
CM-CEUP-2011-V	16	34	5	5	382	400	169	M
MGA-CEJXXIII-2010-V	16	35	5	5	309	400	192	M
NE-CESVP-2011-V-A	16	44	5	5	181	400	183	M
NE-CESVP-2011-V-B	16	43	5	5	192	400	184	M
NE-CESVP-2011-V-C	16	43	5	5	218	400	182	M
NE-CESVP-2011-M-A	18	45	5	5	156	450	212	M
NE-CESVP-2011-M-B	18	44	5	5	167	450	212	M
NE-CESVP-2011-M-C	18	45	5	5	152	450	211	M
NE-CESVP-2011-M-D	18	45	5	5	267	450	211	M
MGA-CEDC-2011-M	19	37	5	5	382	475	210	M
CM-CECM-2011-M	20	51	5	5	648	500	234	M
MGA-CEGV-2011-M	31	62	5	5	588	775	352	L
MGA-CEGV-2011-V	32	75	5	5	857	800	357	L

In this experiment, we analyze two cases. In Section 5.2.1, we compare the results of our algorithms with lower bounds computed by the cut and column generation algorithm of Santos et al. [10]. In Section 5.2.2, we consider a slight modification of the problem and compare our approach with the solver winner of the ITC2011.

5.2.1. Comparison with lower bounds computed by cut and column generation [10]

This section compares the results of our algorithms with lower bounds computed by the CCG algorithm proposed by Santos et al. [10]. For this experiment, the proposed algorithms were set with time limit $t_{max} = 600$ seconds. The results are shown in Table 4. The column LB shows the lower bounds computed by the CCG algorithm. The asterisk (*) means an optimal solution compared to the lower bounds. The best solutions are shown in

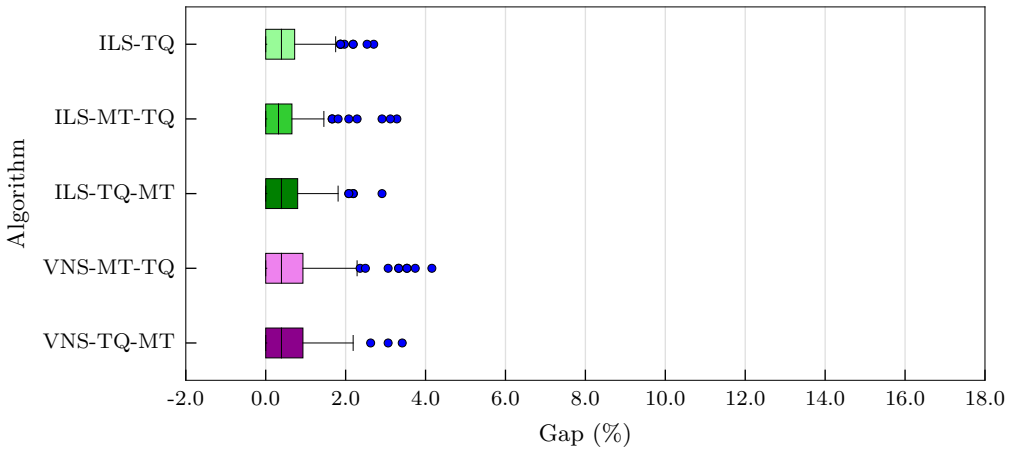
bold font. The character (–) means that the CCG algorithm did not converge in a maximum computational time of 24 hours. The column Gap is the gap between our best solution and the lower bound for instances in which the CCG converged.

Table 4: Best results of our algorithms compared to lower bounds of the CCG in the new instances.

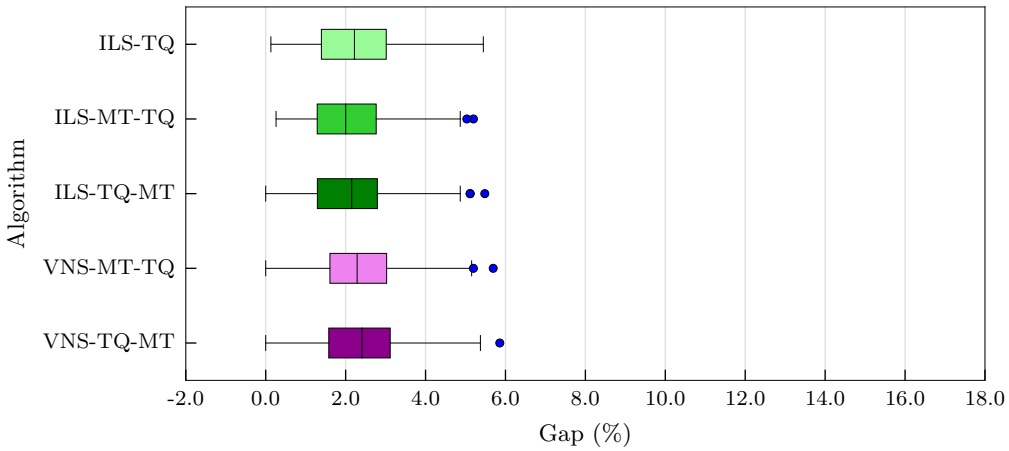
			ILS			VNS		
Instance	Size	LB	TQ	MT-TQ	TQ-MT	MT-TQ	TQ-MT	Gap (%)
CM-CEUP-2011-N	S	–	**269**	**269**	**269**	**269**	**269**	
FA-EEF-2011-M	S	254	*	*	*	*	*	0.00
JNS-CEJXXIII-2011-N	S	254	*	*	*	*	*	0.00
CM-CEDB-2010-N	S	298	*	*	*	*	*	0.00
JNS-CEJXXIII-2011-M	S	319	*	*	*	*	*	0.00
JNS-CEJXXIII-2011-V	S	322	**323**	**323**	**323**	**323**	**323**	0.31
JNS-CEDPII-2011-V	S	457	458	*	458	458	458	0.00
CM-CECM-2011-N	S	–	668	667	**664**	672	670	
JNS-CEDPII-2011-M	S	481	482	*	*	482	*	0.00
CL-CECL-2011-N-A	S	625	**627**	**627**	**627**	**627**	**627**	0.32
MGA-CEVB-2011-V	S	552	*	553	*	554	553	0.00
MGA-CEVB-2011-M	S	–	**571**	573	573	573	574	
CL-CEASD-2008-V-A	M	697	703	**701**	704	703	**701**	0.57
CL-CEASD-2008-V-B	M	697	703	**700**	701	703	703	0.43
MGA-CEDC-2011-V	M	–	728	728	**727**	**727**	728	
CL-CECL-2011-M-A	M	742	744	744	**743**	746	745	0.13
CL-CECL-2011-M-B	M	734	**737**	738	738	738	738	0.41
CM-CECM-2011-V	M	–	**808**	810	811	810	810	
CL-CECL-2011-V-A	M	772	773	774	*	*	*	0.00
CM-CEUP-2008-V	M	–	980	979	**976**	980	986	
CM-CEUP-2011-M	M	–	1030	1030	**1027**	1033	1032	
CM-CEUP-2011-V	M	933	**938**	940	940	942	946	0.54
MGA-CEJXXIII-2010-V	M	912	928	**925**	928	928	929	1.43
NE-CESVP-2011-V-A	M	1010	1030	**1025**	1026	1028	1030	1.49
NE-CESVP-2011-V-B	M	1016	1033	**1030**	1031	1031	1035	1.38
NE-CESVP-2011-V-C	M	1003	1021	1022	**1020**	1024	1021	1.69
NE-CESVP-2011-M-A	M	1121	**1131**	1135	1133	1137	1141	0.89
NE-CESVP-2011-M-B	M	1109	**1124**	1126	1130	**1124**	1131	1.35
NE-CESVP-2011-M-C	M	–	1147	**1144**	1150	1149	1149	
NE-CESVP-2011-M-D	M	1126	1145	1143	1143	1151	**1142**	1.42
MGA-CEDC-2011-M	M	1049	1062	**1060**	1061	1063	1064	1.05
CM-CECM-2011-M	M	1212	1252	1245	**1239**	1245	1250	2.23
MGA-CEGV-2011-M	L	–	1884	1879	1877	**1868**	1886	
MGA-CEGV-2011-V	L	–	2055	**2054**	2058	2064	2064	

Table 4 shows that our best solutions are very close to optimal solutions in general. For small instances, the algorithms were able to find optimal or very close to optimal solutions. However, as the instances' size increases, the gaps enlarge. This is also observed in the distribution of solutions' optimality gaps shown in Figure 5. This figure shows the statistical distribution of the solutions' optimality gaps, in 25 trials, for those instances with lower bounds available. Figure 5(a) shows that our algorithms consistently generated near-optimal solutions for small instances. On the other hand, for medium instances, Figure 5(b) shows that the quality decreased and the gaps became a bit larger, but still in an acceptable tolerance, between 2 and 3 % in the median case. These results show that these new problem

instances are harder to be solved than the instances of Souza et al. [14].



(a) Small instances.



(b) Medium instances.

Figure 5: Distribution of optimality gaps for solutions found by our algorithms in the new instances.

5.2.2. Comparison with the winner of the ITC2011

This section compares the simplest version of our algorithms (the ILS-TQ) with the solver GOAL [17], that is the winner of the ITC2011. In this experiment, we consider a variant of the problem described in Section 2. The new problem has an additional hard constraint to ensure that "hc_6*: the daily schedules of each requirement* $r \in R$ *are always consecutive*". This is one of the high school timetabling problem variants considered in the ITC2011.

For the additional hard constraint hc_6, we used the penalty parameter $\alpha_6^{hc} = 5.000$. The corresponding violations are quantified by β_6^{hc}, computed as follows:

$\beta_6^{hc} = \sum_{r=1}^{|R|} \sum_{d \in D} \omega_{rd}$, where ω_{rd} is the number of holes occurring in the schedule of the r-th requirement within day d. A *hole* is a period (without assignment) which splits the daily

Table 5: The best results of ILS-TQ compared to the best results of GOAL in 25 trials.

Instance	Size	ILS-TQ	GOAL
CM-CEUP-2011-N	S	**273**	**273**
FA-EEF-2011-M	S	Infeasible	
JNS-CEJXXIII-2011-N	S	**254**	**254**
CM-CEDB-2010-N	S	**298**	**298**
JNS-CEJXXIII-2011-M	S	**319**	321
JNS-CEJXXIII-2011-V	S	**325**	328
JNS-CEDPII-2011-V	S	**458**	466
CM-CECM-2011-N	S	687	**676**
JNS-CEDPII-2011-M	S	**482**	491
CL-CECL-2011-N-A	S	**631**	641
MGA-CEVB-2011-V	S	**554**	567
MGA-CEVB-2011-M	S	**574**	591
CL-CEASD-2008-V-A	M	**712**	725
CL-CEASD-2008-V-B	M	**707**	735
MGA-CEDC-2011-V	M	Infeasible	
CL-CECL-2011-M-A	M	**747**	761
CL-CECL-2011-M-B	M	**741**	762
CM-CECM-2011-V	M	**819**	834
CL-CECL-2011-V-A	M	**774**	801
CM-CEUP-2008-V	M	**995**	1042
CM-CEUP-2011-M	M	**1039**	1040
CM-CEUP-2011-V	M	**948**	964
MGA-CEJXXIII-2010-V	M	**937**	973
NE-CESVP-2011-V-A	M	**1040**	1074
NE-CESVP-2011-V-B	M	**1043**	1074
NE-CESVP-2011-V-C	M	**1039**	1070
NE-CESVP-2011-M-A	M	**1146**	1189
NE-CESVP-2011-M-B	M	**1140**	1179
NE-CESVP-2011-M-C	M	**1157**	1187
NE-CESVP-2011-M-D	M	**1153**	1197
MGA-CEDC-2011-M	M	**1069**	1092
CM-CECM-2011-M	M	**1262**	1301
MGA-CEGV-2011-M	L	**1907**	2008
MGA-CEGV-2011-V	L	**2068**	2163
Rank:		31	4

schedule of a requirement in non-consecutive assignments. For example, if a requirement is assigned only to the first and fourth periods of a day, it is counted two holes in this day.

In this comparison, the ILS-TQ is compared with the latest version of GOAL [18]. Both algorithms were executed in the same machine. The solver GOAL was set with suggested parameters (alg-timelimit = 90, form-timelimit = 90, initial-soln = KHE, algorithm = SVNS, formulation = FIXOPT, formfixopt-nresources = 5 and formfixopt-optinarow = 5) and executed with a single thread.

The results are shown in Table 5[5]. The table presents the best results of each algorithm for 25 trials of 900 seconds each. The best solutions are shown in bold font. Comparing the

[5]For this new problem, the instances FA-EEF-2011-M and MGA-CEDC-2011-V were certified to be infeasible by using a simplified MIP program only with feasibility constraints.

best solutions, the algorithm ILS-TQ performed better than GOAL in almost every instance. It ranked 31 best solutions, against 4 of GOAL. Our algorithm only performed worse than GOAL in one instance.

Figure 6 shows the distribution of gaps for solutions found by each algorithm in 25 trials. The gap values are relative to the best-known solutions presented in Table 5. The box plot of Figure 6 shows that the algorithm ILS-TQ is more likely to obtain better quality solutions than GOAL. Moreover, it is more robust than GOAL, as it presented less scattered solutions.

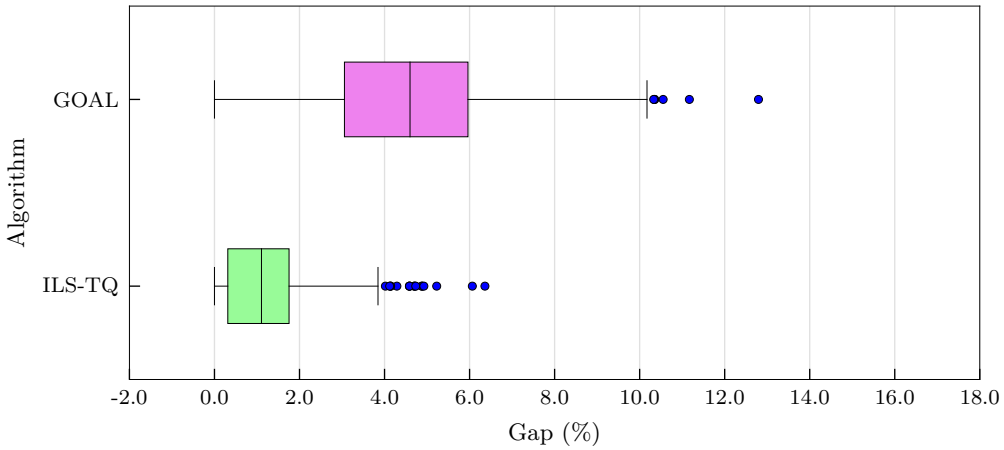


Figure 6: Distribution of gaps for solutions found by the ILS-TQ compared to solutions of GOAL in 25 trials. The gap values are relative to the best-known solutions presented in Table 5.

6. Conclusions

This paper proposed an ILS/VNS soft computing approach to solve high school timetabling problems. The algorithms switch between two local search procedures to explore the solution space. The neighborhoods of the current solution are efficiently and effectively explored by hybridizing two neighborhood operators that are specialized in the problem's structure.

We tested the proposed approach with two variants of the high school timetabling problem. The experimental results showed that our algorithms consistently provided optimal and near-optimal solutions for instances of the problem when compared with lower bounds generated by column generation. Moreover, the proposed algorithms outperformed state-of-the-art approaches in a literature benchmark, and the simplest version of these algorithms outperformed the solver winner of the ITC2011 in a large number of real-world instances introduced in this paper.

This study demonstrated the efficiency, effectiveness, and robustness of our approach compared to literature approaches. The proposed algorithms and neighborhood operators are easily adaptable to other high school timetabling problems. Future works will focus on embedding these algorithms into parallel metaheuristic frameworks to further exploit their power.

Acknowledgments

This research was supported by FAPESP (Grant n. 2013/13563-3); Fundação Araucária (Grant n. 207/2014); and CAPES Brazil. The authors would like to thank professor George H. G. Fonseca for sharing the code of GOAL, professor Haroldo Gambini Santos for sharing his cut and column generation algorithm, and the anonymous reviewers for their useful comments and suggestions to improve this paper.

References

[1] R. Lewis, A survey of metaheuristic-based techniques for university timetabling problems, OR Spectrum 30 (2008) 167–90.

[2] R. Qu, E. K. Burke, B. McCollum, L. T. Merlot, S. Y. Lee, A survey of search methodologies and automated system development for examination timetabling, Journal of scheduling 12 (2009) 55–89.

[3] N. Pillay, W. Banzhaf, An informed genetic algorithm for the examination timetabling problem, Applied Soft Computing 10 (2010) 457–67.

[4] D. De Werra, An introduction to timetabling, European Journal of Operational Research 19 (1985) 151–62.

[5] G. Post, S. Ahmadi, S. Daskalaki, J. Kingston, J. Kyngas, C. Nurmi, D. Ranson, An xml format for benchmarks in high school timetabling, Annals of Operations Research 194 (2012) 385–97.

[6] N. Pillay, A survey of school timetabling research, Annals of Operations Research 218 (2014) 261–93.

[7] D. De Werra, Construction of school timetables by flow methods, INFOR: Information Systems and Operational Research 9 (1971) 12–22.

[8] S. Even, A. Itai, A. Shamir, On the complexity of timetable and multi-commodity flow problems, in: 16th Annual Symposium on Foundations of Computer Science, IEEE, pp. 184–93.

[9] T. B. Cooper, J. H. Kingston, The complexity of timetable construction problems, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 281–95.

[10] H. Santos, E. Uchoa, L. Ochi, N. Maculan, Strong bounds with cut and column generation for class-teacher timetabling, Annals of Operations Research 194 (2012) 399–412.

[11] S. Kristiansen, M. Sørensen, T. R. Stidsen, Integer programming for the generalized high school timetabling problem, Journal of Scheduling 18 (2015) 377–92.

[12] Á. P. Dorneles, O. C. de Araújo, L. S. Buriol, A column generation approach to high school timetabling modeled as a multicommodity flow problem, European Journal of Operational Research 256 (2017) 685–95.

[13] G. H. Fonseca, H. G. Santos, E. G. Carrano, T. J. Stidsen, Integer programming techniques for educational timetabling, European Journal of Operational Research (2017).

[14] M. J. F. Souza, N. Maculan, L. S. Ochi, A GRASP-Tabu Search Algorithm for Solving School Timetabling Problems, Springer US, pp. 659–72.

[15] I. X. Tassopoulos, G. N. Beligiannis, A hybrid particle swarm optimization based algorithm for high school timetabling problems, Applied Soft Computing 12 (2012) 3472–89.

[16] Á. P. Dorneles, O. C. de Araújo, L. S. Buriol, A fix-and-optimize heuristic for the high school timetabling problem, Computers & Operations Research 52, Part A (2014) 29–38.

[17] G. H. G. Fonseca, H. G. Santos, T. Â. M. Toffolo, S. S. Brito, M. J. F. Souza, Goal solver: a hybrid local search based solver for high school timetabling, Annals of Operations Research 239 (2016) 77–97.

[18] G. H. Fonseca, H. G. Santos, E. G. Carrano, Integrating matheuristics and metaheuristics for timetabling, Computers & Operations Research 74 (2016) 108–17.

[19] V. I. Skoullis, I. X. Tassopoulos, G. N. Beligiannis, Solving the high school timetabling problem using a hybrid cat swarm optimization based algorithm, Applied Soft Computing 52 (2017) 277–89.

[20] H. Santos, L. Ochi, M. Souza, A tabu search heuristic with efficient diversification strategies for the class/teacher timetabling problem, Journal of Experimental Algorithmics 10 (2005) 2–9.

[21] C. Valouxis, E. Housos, Constraint programming approach for school timetabling, Computers & Operations Research 30 (2003) 1555–72.

[22] P. Avella, B. D'Auria, S. Salerno, I. Vasil'ev, A computational study of local search algorithms for italian high-school timetabling, Journal of Heuristics 13 (2007) 543–56.

[23] T. Birbas, S. Daskalaki, E. Housos, School timetabling for quality student and teacher schedules, Journal of scheduling 12 (2009) 177–97.

[24] J. S. Appleby, D. V. Blake, E. A. Newman, Techniques for producing school timetables on a computer and their application to other scheduling problems, The Computer Journal 3 (1961) 237–45.

[25] J. Csima, C. C. Gotlieb, Tests on a computer method for constructing school timetables, Commun. ACM 7 (1964) 160–3.

[26] G. N. Beligiannis, C. N. Moschopoulos, G. P. Kaperonis, S. D. Likothanassis, Applying evolutionary computation to the school timetabling problem: The greek case, Computers & Operations Research 35 (2008) 1265–80.

[27] G. N. Beligiannis, C. Moschopoulos, S. D. Likothanassis, A genetic algorithm approach to school timetabling, Journal of the Operational Research Society 60 (2009) 23–42.

[28] D. Zhang, Y. Liu, R. M'Hallah, S. C. Leung, A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems, European Journal of Operational Research 203 (2010) 550–8.

[29] G. Post, J. H. Kingston, S. Ahmadi, S. Daskalaki, C. Gogos, J. Kyngas, C. Nurmi, N. Musliu, N. Pillay, H. Santos, A. Schaerf, Xhstt: an xml archive for high school timetabling problems in different countries, Annals of Operations Research 218 (2014) 295–301.

[30] G. Post, L. Di Gaspero, J. H. Kingston, B. McCollum, A. Schaerf, The third international timetabling competition, Annals of Operations Research 239 (2016) 69–75.

[31] H. R. Lourenço, O. C. Martin, T. Stützle, Iterated Local Search, Springer US, Boston, MA, pp. 320–53.

[32] P. Hansen, N. Mladenović, J. Moreno Pérez, Variable neighbourhood search: methods and applications, Annals of Operations Research 175 (2010) 367–407.

[33] A. A. Constantino, D. Landa-Silva, E. L. de Melo, C. F. X. de Mendonça, D. B. Rizzato, W. Romão, A heuristic algorithm based on multi-assignment procedures for nurse scheduling, Annals of Operations Research 218 (2014) 165–83.

[34] Z. Lü, J.-K. Hao, F. Glover, Neighborhood analysis: a case study on curriculum-based course timetabling, Journal of Heuristics 17 (2011) 97–118.

[35] G. Carpaneto, P. Toth, Primal-dual algrorithms for the assignment problem, Discrete Applied Mathematics 18 (1987) 137–53.

Highlights

- We propose Iterated Local Search and Variable Neighborhood Search based algorithms for high school timetabling problems.
- We devise two generic neighborhood operators that can be easily adapted to similar problems.
- The hybridization between these neighborhood operators improves hugely the effectiveness of local search procedures.
- The proposed algorithms outperform state-of-the-art approaches in two datasets tested for two variants of the problem.
- New instances collected from different Brazilian high schools are made available for next studies.