Pattern-based models and a cooperative parallel metaheuristic for high school timetabling problems

Landir Saviniec[a], Maristela O. Santos[b], Alysson M. Costa[c] and Lana M. R. dos Santos[d]

[a] Universidade Federal do Paraná, Brazil
[b] Universidade de São Paulo, Brazil
[c] The University of Melbourne, Australia
[d] Universidade Federal de Viçosa, Brazil

October 31, 2018

Abstract

High school timetabling problems consist in building periodic timetables for class-teacher meetings considering compulsory and non-compulsory requisites. This family of problems has been widely studied since the 1950s, mostly via mixed-integer programming and metaheuristic techniques. However, the efficient obtention of optimal or near-optimal solutions is still a challenge for many problems of practical size. In this paper, we investigate mixed-integer programming formulations and a parallel metaheuristic based algorithm for solving high school timetabling problems with compactness and balancing requisites. We propose two pattern-based formulations and a solution algorithm that simultaneously exploits column generation and a team of metaheuristics to build and improve solutions. Extensive computational experiments conducted with real-world instances demonstrate that our formulations are competitive with the best existing high school timetabling formulations, while our parallel algorithm presents superior performance than state-of-the-art methods.

Keywords: Class-teacher timetabling, Parallel metaheuristics, Column Generation, Iterated local search.

1 Introduction

Educational timetabling problems consist in assigning meetings between teachers (or exams) and students, considering a list of different requisites. Usually, these requisites are of two types: hard (mandatory) or soft (non-mandatory). A timetable that satisfies all hard requisites is said to be feasible, and a timetable that is feasible and also has a minimum number of violated soft requisites is said to be optimal.

The automated design of educational timetables started in the late 1950s, with methods that tried to encode manual procedures. Junginger (1986) presents a review of the first papers on timetabling problems, showing that simple heuristics were the most common solution methods. The article also reports on early computer programs that had been developed using exact methods (Gotlieb, 1963). However, as expected, they were able to address only small instances of simplified problems. Since then, the timetabling literature has been widely extended. Current research can be categorized around three main educational timetabling variants: university course timetabling (Lewis, 2008), examination timetabling (Qu et al., 2009) and high school timetabling (De Werra, 1985; Pillay, 2014).

In this paper, we address the high school timetabling problem (HSTP). In its most basic version, which is NP-complete (Even et al., 1975), the HSTP only requires that scheduling clashes and unavailability of teachers are avoided. Here, we consider an HSTP variant based on Brazilian schools common requirements, such as: enforcing double lessons for some classes' subjects (two consecutive meetings between a teacher and a class within the same day), avoiding idle periods (a free period between busy periods within the same day in teachers' schedules), and prioritizing compact schedules for teachers. An ideal *compact* schedule is the one in which a teacher works the minimum number of days to meet his/her teaching requirements. Any worked days above this minimum amount are called *extra working days*, and should be balanced among the teachers.

Traditional compact mixed-integer programming (MIP) models (Dorneles et al., 2014; Sørensen and Dahms, 2014; Kristiansen et al., 2015) for high school timetabling problems are known to have weak linear programming relaxations. Stronger MIP formulations have been proposed (Dorneles et al., 2017; Fonseca et al., 2017). However, these formulations are still ineffective when implemented to solve medium and large size instances using current black-box solvers. Given these limitations, many researchers have focused on combining heuristic algorithms and relaxation methods to obtain high-performance hybrid approaches and optimality bounds. On the one hand, research on heuristic algorithms aims the computation of high-quality solutions with reduced computational effort. On the other hand, the research on relaxation methods aims the computation of tight lower bounds on the optimal solutions.

The quality of the lower bounds obtained by compact formulations can sometimes be improved by solving the linear relaxation of extended MIP formulations via column generation. Indeed, some extended formulations proposed to HSTPs have shown strong lower bounds (Papoutsis et al., 2003; Santos et al., 2012; Dorneles et al., 2017). Simultaneously, recent heuristic approaches (Saviniec and Constantino, 2017; Skoullis et al., 2017; Saviniec et al., 2018) have found good quality solutions. Even with these recent advances, optimality gaps for medium and large size available instances are still not closed and motivate further research in both the development of good lower and upper bounds.

We propose two new MIP formulations for the addressed HSTP variant. The first formulation has a relatively small number of variables when applied to realistic instances. This formulation is strengthened using Fenchel cuts (Boyd, 1994b, a) and it is explicitly generated and solved with a black-box solver. The second formulation has an enormous

number of variables, even for small-size problems, and we therefore resort to column generation to solve its linear relaxation.

We also propose a cooperative parallel solution algorithm. The method uses a team of metaheuristics to construct and improve solutions. The solution algorithm also incorporates new agents based on the previously mentioned column generation. These agents use partial/fractional solutions obtained by the column generation to i) obtain lower bounds for the problem and ii) extend them to complete solutions using an original method that relies on a fix-and-optimize heuristic with an inbuilt factibilisation procedure based on proximity search.

To assess the proposed approaches, we conduct a thorough computational study with instances of the HSTP variant of interest. The methods are also adapted to address two related versions of the problem. The remainder of this paper is organized as follows. Section 2 presents a literature review of state-of-the-art techniques for high school timetabling. Section 3 formally defines the problem under study. Section 4 is dedicated to the presentation of the proposed MIP models, while Section 5 details our solution algorithm. Section 6 provides the computational experiments and analysis, and Section 7 presents final remarks and suggestions for future research. Two appendices complete this article. Appendix A presents an additional MIP model (Saviniec et al., 2018) that is also used within our solution algorithm. Appendix B provides details on the stand-alone metaheuristic methods that are also employed by the proposed parallel algorithm.

2 State-of-the-art techniques for HSTP

Since the first studies in the late 1950s, the scientific literature has reported numerous papers dealing with solution techniques for high school timetabling problems. In this section, we review the most successful of these approaches.

Up to the 2000s, the high school timetabling research had been mostly based on independent case studies, making it difficult to compare different solution methods. The first high school timetabling dataset to be widely available appeared in the early 2000s. Souza (2000) introduced a small set of instances from Brazilian high schools[1], which became a commonly used benchmark, being addressed later by both heuristics (Souza et al., 2004; Santos et al., 2005; Dorneles et al., 2014; Saviniec and Constantino, 2017; Saviniec et al., 2018) and mixed-integer programming techniques (Santos et al., 2012; Dorneles et al., 2014, 2017). This dataset was extended with new real cases[2] proposed by Saviniec and Constantino (2017). The state-of-the-art heuristic methods for solving this new problem are parallel metaheuristic-based algorithms (Saviniec et al., 2018).

A second high school timetabling dataset appeared in the late 2000s. Beligiannis et al. (2008) introduced a set of real-case instances from Greek high schools[3] that also became the subject of several studies (Beligiannis et al., 2009; Zhang et al., 2010; Tassopoulos and Beligiannis, 2012; Skoullis et al., 2017). The state-of-the-art heuristic method reported for this dataset is a Cat Swarm Optimization based algorithm (Skoullis et al., 2017).

During the The 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT, 2008), the timetabling community proposed an XML-based format to represent different high school timetabling problems and create a unified set of benchmark instances (Post et al., 2012). This format, called XHSTT (Post et al., 2014), established rules to model the most common high school timetabling requisites found

[1] Available in `http://labic.ic.uff.br/Instance/`

[2] Available in `http://www.gpea.uem.br/benchmark.html`

[3] Available in `http://www.deapt.upatras.gr/cso\_timetabling/school-timetabling.html`

around the world. The XHSTT format allows the description of *generalized high school timetabling problems* (GHSTP). After its creation, new instances from different countries have been added to the XHSTT repository[4], including some of those proposed by Souza (2000). Such instances were used in the Third International Timetabling Competition (ITC-2011) devoted to HSTPs. The state-of-the-art method for these instances is the heuristic solver GOAL (Fonseca et al., 2016b), which won the ITC-2011. An updated version of GOAL is described in Fonseca et al. (2016a).

Regarding the use of mixed-integer programming techniques, different compact formulations have been proposed to HSTPs (Santos et al., 2012; Dorneles et al., 2014; Sørensen and Dahms, 2014; Al-Yakoob and Sherali, 2015; Kristiansen et al., 2015). However, all of these formulations have shown weak linear relaxations. The strongest formulation for high school timetabling is based on a multi-commodity flow problem, first proposed for problems of Brazilian high schools (Dorneles et al., 2017) and later adapted to solve GHSTP (Fonseca et al., 2017). Extended formulations solved by column generation have also been proposed. Santos et al. (2012) and Dorneles et al. (2017) proposed different formulations for the Brazilian case, which showed similar results regarding the quality of their lower bounds. Finally, a column generation lower bounding procedure was also proposed in Al-Yakoob and Sherali (2015), for problems originated at Kuwaiti high schools.

As mentioned before, the literature has reported many other case studies dealing with solution techniques for high school timetabling problems. For a more detailed review of these methods, we refer the interested reader to Pillay (2014).

3 Problem definition

We consider the high school timetabling problem studied in Saviniec et al. (2018), which is originated from the context of Brazilian public high schools. The problem structure is represented in Figure 1. The school has a set of classes C and a set of teachers T. Classes are disjoint groups of students enrolled in the same list of subjects (e.g., mathematics, physics, etc.) and with no free periods during the week. Each class' subject has a number of weekly meetings (or lessons) that are taught by a preassigned teacher. The set of all classes' subjects is called "Required meetings" (R) and each class' subject is called "a requirement". Teachers may be unavailable in some periods/days. Given these input data, the goal is to obtain weekly timetable specifying the schedule for all required meetings in a particular shift (morning, afternoon, or evening). A timetable is constructed for each independent shift, and it usually spreads over the set of weekdays D (Monday to Friday) with five periods each. We define H as the set of periods per day and refer to periods as a set of timeslots $(d, h) \in D \times H$.

The timetabling process must fulfill the following hard requisites:

1. **Meeting of weekly required lessons**: all required lessons must be assigned.
2. **No clashes in classes' schedules**: each class $c \in C$ must attend exactly one lesson per period.
3. **No clashes in teachers' schedules**: each teacher $t \in T$ must teach at most one lesson per period.
4. **No assignment of teachers in their unavailable periods**: teachers must not be assigned to periods in which they are unavailable.

[4]Available in `https://www.utwente.nl/ctit/hstt/`

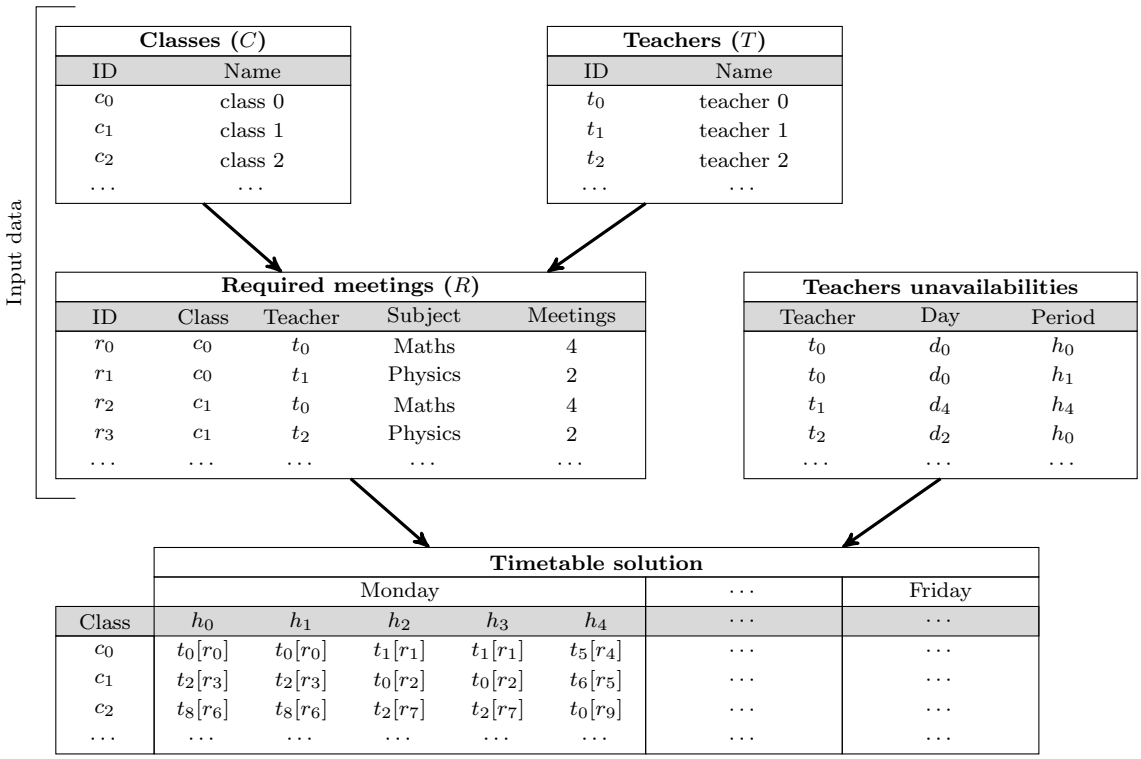


Figure 1: Example of high school timetabling data and solution.

The following soft requisites are also considered:

5. **No daily workload violation for requirements**: for each requirement $r \in R$, there is a limit $\widetilde{\delta}_r$ on the amount of meetings per day.

6. **No gaps in requirements' schedules**: gaps in requirements' schedules should be avoided. A *gap* is a period splitting the daily schedule of a requirement into non-consecutive meetings.

7. **Meeting of double lessons for requirements**: for each requirement $r \in R$, there is a specified minimum number of consecutive double lessons $\widetilde{\pi}_r$ that should be met.

8. **No idle periods in teachers' schedules**: teachers should not have idle periods in their daily schedules. In this problem, unavailable periods are not computed as idle periods.

9. **Compact schedules for teachers**: teachers should be scheduled to come to school the minimum possible number of days.

10. **Balancing on teachers' extra working days**: the extra working days should be balanced among teachers. The number of extra working days for a teacher $t \in T$ is the difference between the number of worked days and an estimated lower bound $\widetilde{d}_t$, defined later in Table 1.

4 Mixed integer programming models

This section presents our two models for the problem. Both models use variables representing 'layouts', i.e., possible (partial) schedules of teachers/requirements. Both models make use of parameters described in Table 1. Details of the models are presented in the following subsections.

Table 1: Input parameters of a high school timetabling instance.

Notation	Definition
Sets	
C	the set of classes.
T	the set of teachers.
D	the set of weekdays.
H	the set of periods per day.
$H_{td} \subseteq H$	the subset of periods in day $d \in D$ for which teacher $t \in T$ is available.
R	the set of required meetings (or lessons), as defined in Section 3.
$R_c \subseteq R$	the subset of requirements associated to a class $c \in C$.
$R_t \subseteq R$	the subset of requirements associated to a teacher $t \in T$.
Parameters	
$\widetilde{\theta}_r \in \mathbb{N}$	the number of weekly lessons for a requirement $r \in R$.
$\widetilde{\delta}_r \in \mathbb{N}$	the daily limit for the number of lessons of a requirement $r \in R$.
$\widetilde{\pi}_r \in \mathbb{N}$	the desired minimum number of double lessons for a requirement $r \in R$.
$\widetilde{d}_t = \left\lceil \frac{\sum_{r \in R_t} \widetilde{\theta}_r}{\lvert H \rvert} \right\rceil$	a lower bound on the minimum number of days for which a teacher $t \in T$ can be assigned and still meet his/her teaching requirements.

Table 2: Additional notation and decision variables used in formulation EF1.

Notation	Definition
Sets	
L	the set of assignment layouts that a requirement r (or teacher t) can follow in a day.
Parameters	
$\widetilde{\mu}_{lh} \in \{0,1\}$	indicates whether a layout $l \in L$ has an assignment at period $h \in H$.
$\widetilde{a}_l \in \mathbb{N}$	the number of assignments in a layout l.
$\widetilde{e}_{rl} \in \mathbb{N}$	the number of lessons that exceed the daily limit $\widetilde{\delta}_r$ in a layout l for a requirement r.
$\widetilde{h}_l \in \mathbb{N}$	the number of gaps in a layout l for requirements.
$\widetilde{\omega}_l \in \mathbb{N}$	the number of consecutive double lessons in a layout l for requirements.
$\widetilde{\phi}_l \in \{0,1\}$	indicates whether the number of assignments in a layout l is greater than zero.
$\widetilde{i}_{tdl} \in \mathbb{N}$	the number of idle periods in a layout l for a teacher t in a day d.
Decision variables	
$x_{rdl} \in \{0,1\}$	indicates whether a requirement r follows a layout l in a day d.
$y_{tdl} \in \{0,1\}$	indicates whether a teacher t follows a layout l in a day d.
Auxiliary variables	
$\widehat{\pi}_r \in \mathbb{N}$	the number of unmet weekly double lessons for a requirement r.
$\widehat{\beta} \in \mathbb{N}$	the largest value among all teachers' extra working days.

4.1 Extended formulation 1

The first model (EF1) uses the idea of assignment layouts (*patterns*) for the daily schedules of teachers and requirements. A *daily assignment layout* is defined as a binary vector of size $|H|$ in which the h^{th} position is associated with the h^{th} period of the day. For a teacher, a bit value of *1* indicates that the teacher is assigned at that period, while for a requirement it indicates that there is one meeting associated with that requirement at that period.

The number of all possible patterns for a requirement or teacher (disregarding unavailable periods) is given by $|D| \cdot 2^{|H|}$. For $|H| = 5$, a teaching layout '00000' indicates that the associated teacher has a day off while a layout '00111' indicates that the teacher has classes assigned to him/her in the last three teaching periods of the day. This definition of "layouts" is different from most literature that usually explicitly codify which meetings

are happening. Here, for teacher patterns, there is only the definition of the presence or absence of the considered teacher in a timeslot. For requirement patterns, a layout '11000' indicates a double lesson in the first two periods while a layout '10100' represents an undesired gap between two classes of the same requirement.

Table 2 presents the data of the HSTP in an appropriate format and also lists the two sets of binary decision variables, x_{rdl} and y_{tdl}. These binary variables indicate whether a requirement $r \in R$ (x_{rdl}) or teacher $t \in T$ (y_{tdl}) follows an assignment pattern $l \in \{0, 1, \cdots, 2^{|H|} - 1\}$ on day $d \in D$. Model EF1 can thus be written as:

$$\textbf{Minimize} \quad \sum_{r\in R}\sum_{d\in D}\sum_{l\in L}(\alpha_5\widetilde{e}_{rl} + \alpha_6\widetilde{h}_l)x_{rdl} + \alpha_7\sum_{r\in R}\widehat{\pi}_r + \sum_{t\in T}\sum_{d\in D}\sum_{l\in L}(\alpha_8\widetilde{i}_{tdl} + \alpha_9\widetilde{\phi}_l)y_{tdl} + \alpha_{10}\widehat{\beta} \tag{1}$$

Subject to:

$$\sum_{d\in D}\sum_{l\in L}\widetilde{a}_l x_{rdl} = \widetilde{\theta}_r \qquad \forall\ r \in R \tag{2}$$

$$\sum_{r\in R_c}\sum_{l\in L}\widetilde{\mu}_{lh}x_{rdl} = 1 \qquad \forall\ c \in C;\ d \in D;\ h \in H \tag{3}$$

$$\sum_{l\in L} x_{rdl} = 1 \qquad \forall\ r \in R;\ d \in D \tag{4}$$

$$\sum_{l\in L} y_{tdl} = 1 \qquad \forall\ t \in T;\ d \in D \tag{5}$$

$$\widehat{\pi}_r \geq \widetilde{\pi}_r - \sum_{d\in D}\sum_{l\in L}\widetilde{\omega}_l x_{rdl} \qquad \forall\ r \in R \tag{6}$$

$$\widehat{\beta} \geq \sum_{d\in D}\sum_{l\in L}\widetilde{\phi}_l y_{tdl} - \widetilde{d}_t \qquad \forall\ t \in T \tag{7}$$

$$\sum_{l\in L}\widetilde{\mu}_{lh}y_{tdl} = \sum_{r\in R_t}\sum_{l\in L}\widetilde{\mu}_{lh}x_{rdl} \qquad \forall\ t \in T;\ d \in D;\ h \in H \tag{8}$$

$$\widehat{\pi}_r \geq 0 \qquad \forall\ r \in R \tag{9}$$

$$\widehat{\beta} \geq 0 \tag{10}$$

$$x_{rdl} \in \{0, 1\} \qquad \forall\ r \in R;\ d \in D;\ l \in L \tag{11}$$

$$y_{tdl} \in \{0, 1\} \qquad \forall\ t \in T;\ d \in D;\ l \in L \tag{12}$$

Objective function (1) minimizes violations of soft requisites **5** to **10**, in which α_i is a penalty parameter associated with the relative importance of the i^{th} requisite. Constraints (2) ensure that all required lessons are scheduled. Constraints (3) avoid clashes in classes' schedules. Constraints (4) and (5) ensure that a single layout is assigned to each requirement and each teacher on a given day, respectively. Constraints (6) and (7) link the auxiliary variables to the decision variables in order to quantify violations of soft requisites **7** and **10**: constraints (6) compute the number of unmet double lessons for each requirement while constraints (7) capture the largest value among all teachers' extra working days. Finally, the linking constraints (8) ensure that the daily schedules of teachers match their requirements' schedules.

To avoid assigning teachers to their unavailable periods, we preprocess the associated variables and forbid layouts which contain infeasible assignments. This task can be accomplished by setting the following variables to zero:

$$y_{tdl} = 0 \qquad \forall\ t \in T;\ d \in D;\ l \in L;\ h \in H \setminus H_{td};\ \widetilde{\mu}_{lh} > 0 \tag{13}$$

$$x_{rdl} = 0 \qquad \forall\ t \in T;\ r \in R_t;\ d \in D;\ l \in L;\ h \in H \setminus H_{td};\ \widetilde{\mu}_{lh} > 0 \tag{14}$$

Program (1)–(12) is also augmented with the additional valid cuts (15), which specify that a teacher cannot work less than the minimum necessary number of days $\widetilde{d}_t$ (Souza, 2000).

$$\sum_{d \in D} \sum_{l \in L} \widetilde{\phi}_l y_{tdl} \geq \widetilde{d}_t \qquad \forall\ t \in T \tag{15}$$

We note that since $|H|$ is usually a small constant for most high school timetabling problems (e.g., for our case $|H| = 5$), the number of variables of EF1 tends to remain reasonably low. This allows for black-box MIP solvers to be used for the obtention of solutions. In order to further increase the efficiency of the methods, we strengthen EF1 with valid inequalities, as described in the following.

Table 3: Notation used to define Fenchel cuts to formulation EF1.

Notation	Definition
Sets	
$K_{td} = \{1, \cdots, \lvert H_{td}\rvert\}$	the set of all possible number of lessons that a teacher t can have in a day d.
$Q_{rd} = \{1, \cdots, \min\{\lvert H_{td}\rvert, \widetilde{\theta}_r\}\}$	the set of all possible number of lessons that a requirement r can have in a day d. In this definition, t is the teacher associated with requirement r.
$A_k \subseteq L$	the subset of layouts $l \in L$ which have exactly k bits at value 1, for $k \in \{1, \cdots, \lvert H\rvert\}$.
Additional variables	
$w_{tdk} \in \{0, 1\}$	indicates whether a teacher t has exactly k lessons in a day d.
$v_{rdk} \in \{0, 1\}$	indicates whether a requirement r has exactly k lessons in a day d.

4.1.1 Cutting planes generation for EF1

In this section, we describe cuts that can be added to the root node of program (1)–(12) to strengthen its linear relaxation. The cuts are generated by using Fenchel enumeration cutting planes technique (Boyd, 1994b, a). The idea is adapted from Santos et al. (2012), which employed these cuts to a different formulation of the HSTP problem. In Santos et al. (2012), the cuts are used to strengthen a column generation procedure instead of an explicitly generated model.

Let us consider the notation defined in Table 3. We augment formulation EF1 by introducing auxiliary binary variables w_{tdk} and v_{rdk}. These new variables are linked to the original ones by means of constraints:

$$\sum_{l \in A_k} y_{tdl} = w_{tdk} \qquad \forall\ t \in T;\ d \in D;\ k \in K_{td} \tag{16}$$

$$\sum_{l \in A_k} x_{rdl} = v_{rdk} \qquad \forall\ r \in R;\ d \in D;\ k \in Q_{rd} \tag{17}$$

Now consider each of the following valid polyhedra:

- Polyhedron Π_t (for lessons of a teacher t):

$$\sum_{d\in D}\sum_{k\in K_{td}} k w_{tdk} = \sum_{r\in R_t} \widetilde{\theta}_r \tag{18}$$

$$\sum_{k\in K_{td}} w_{tdk} \leq 1 \qquad \forall\ d \in D \tag{19}$$

$$w_{tdk} \in \{0,1\} \qquad \forall\ d \in D;\ k \in K_{td} \tag{20}$$

Constraints (18) ensure that the weekly workload of teacher t is met, and constraints (19) impose that at most one variable w is active for each day.

- Polyhedron Π_r (for lessons of a requirement r):

$$\sum_{d\in D}\sum_{k\in Q_{rd}} k v_{rdk} = \widetilde{\theta}_r \tag{21}$$

$$\sum_{k\in Q_{rd}} v_{rdk} \leq 1 \qquad \forall\ d \in D \tag{22}$$

$$v_{rdk} \in \{0,1\} \qquad \forall\ d \in D;\ k \in Q_{rd} \tag{23}$$

Constraints (21) ensure that the correct weekly workload of requirement r is met, and constraints (22) impose that at most one variable v is active for each day.

Let $\overline{w}$ be the fractional values of variables w in the optimal solution of the linear program of augmented formulation EF1. By enumerating all integer feasible solutions $\mathring{w} \in \Pi_t$, we can easily check if there is an hyperplane separating $\overline{w}$ from the convex hull defined by the feasible integer points $\mathring{w} \in \Pi_t$. This is done by solving the following oracle linear program $\mathcal{O}_t$:

$$\textbf{Maximize} \quad z_t = \sum_{d\in D}\sum_{k\in K_{td}} \overline{w}_{tdk} a_{tdk} \tag{24}$$

Subject to:

$$\sum_{d\in D}\sum_{k\in K_{td}} \mathring{w}_{tdk} a_{tdk} \leq 1 \qquad \forall\ \mathring{w} \in \Pi_t \tag{25}$$

$$a_{tdk} \geq 0 \qquad \forall\ d \in D;\ k \in K_{td} \tag{26}$$

Let $(z_t^*,\ a_{tdk}^*)$ be the optimal solution of $\mathcal{O}_t$. If $z_t^* > 1$, then point $\overline{w}$ lies outside of Π_t and

$$\sum_{d\in D}\sum_{k\in K_{td}} a_{tdk}^* w_{tdk} \leq 1 \tag{27}$$

is a valid inequality that is violated by the current fractional solution. The exact same idea can be used to obtain violated valid inequalities associated with variables v_{rdk}.

The described inequalities are iteratively identified and inserted into program (1)–(12) at the root node. At each iteration, we solve the linear program of EF1 and call the oracle problems for each teacher and for each requirement. Any violated inequality is inserted in the program. If no new violated inequality is found and the solution is still fractional, then we proceed to branching.

Table 4: Notation and decision variables used in formulation EF2.

Notation	Definition
Sets	
S_t	the set of all possible weekly schedules that a teacher $t \in T$ can follow.
$T_c \subseteq T$	the set of teachers that teach to class $c \in C$.
Parameters	
f_{ti}	the cost associated with violations of requisites 5 to 9 in the i^{th} weekly schedule of a teacher t.
$\rho_{ti}^{cdh} \in \{0,1\}$	indicates whether a teacher t teaches class c in a schedule $i \in S_t$ at period h of a day d.
$\widetilde{\phi}_{ti} \in \mathbb{N}$	the number of working days for a teacher t in a schedule $i \in S_t$.
Decision variables	
$\lambda_{ti} \in \{0,1\}$	indicates whether a teacher t follows a weekly schedule $i \in S_t$.
Auxiliary variables	
$\widehat{\beta} \in \mathbb{N}$	the largest value among all teachers' extra working days.

4.2 Extended formulation 2

This section presents the second extended formulation, denoted by EF2. The variables of EF2 represent complete weekly schedules of teachers. A *weekly schedule* for a teacher $\breve{t}$ is any assignment of binary values to variables $y_{\breve{t}dl}$ and x_{rdl} (for all $r \in R_{\breve{t}}$) in model EF1, that satisfies constraints (2), (4), (5), (6), (8), (13), (14) and (15). We define S_t as the set of all feasible weekly schedules for a teacher $t \in T$. A timetable can thus be represented by a set of binary variables λ_{ti}, which indicate whether each teacher $t \in T$ follows a weekly schedule $i \in S_t$.

Considering the additional notation defined in Table 4, the mixed-integer program of EF2 can be written as:

$$\textbf{Minimize} \quad \sum_{t \in T} \sum_{i \in S_t} f_{ti} \lambda_{ti} + \alpha_{10} \widehat{\beta} \tag{28}$$

Subject to:

$$(\xi_t^1) \qquad \sum_{i \in S_t} \lambda_{ti} = 1 \qquad \forall\ t \in T \tag{29}$$

$$(\xi_{cdh}^2) \qquad \sum_{t \in T_c} \sum_{i \in S_t} \rho_{ti}^{cdh} \lambda_{ti} = 1 \qquad \forall\ c \in C;\ d \in D;\ h \in H \tag{30}$$

$$(\xi_t^3) \qquad \widehat{\beta} - \sum_{i \in S_t} \widetilde{\phi}_{ti} \lambda_{ti} \geq -\widetilde{d}_t \qquad \forall\ t \in T \tag{31}$$

$$\widehat{\beta} \geq 0 \tag{32}$$

$$\lambda_{ti} \in \{0,1\} \qquad \forall\ t \in T;\ i \in S_t \tag{33}$$

Objective function (28) minimizes the cost of violating requisites **5** to **9** in selected weekly teacher schedules, plus the cost associated with soft requisite **10**. Constraints (29) ensure that a single weekly schedule is selected for each teacher. Constraints (30) avoid clashes in classes' schedules, and constraints (31) capture the maximum number of extra working days among all teachers. In this formulation, we also indicate the dual variables associated with constraints (29)–(31) when the integrality constraints (33) are relaxed.

For instances of practical size, formulation EF2 has a huge number of variables that are hard to generate explicitly. Even when the explicit writing of the problem is possible, it usually results in models that cannot be tackled by current solvers. However, as optimal solutions are defined by small subsets of these variables, we can use column generation (Desaulniers et al., 2005) as an effective tool to solve the associated relaxed problem. The procedure is described in the following.

Table 5: Notation used in the formulation of a subproblem $\mathcal{Q}_t$.

Notation	Definition
Sets	
L	the set of all possible assignment layouts that a teacher t or a requirement $r \in R_t$ can follow in a day.
$C_t \subseteq C$	the set of classes taught by teacher t.
$R_{ct} \subseteq R$	the set of requirements belonging to class c which are taught by teacher t.
Parameters	
$\widetilde{\mu}_{lh} \in \{0,1\}$	indicates whether a layout $l \in L$ has an assignment at period $h \in H$.
$\widetilde{a}_l \in \mathbb{N}$	the number of assignments in a layout l.
$\widetilde{e}_{rl} \in \mathbb{N}$	the number of lessons that exceed the daily limit $\widetilde{\delta}_r$ in a layout l for a requirement $r \in R_t$.
$\widetilde{h}_l \in \mathbb{N}$	the number of gaps in a layout l for requirements.
$\widetilde{\omega}_l \in \mathbb{N}$	the number of consecutive double lessons in a layout l for requirements.
$\widetilde{\phi}_l \in \{0,1\}$	indicates whether the number of assignments in a layout l is greater than zero.
$\widetilde{i}_{tdl} \in \mathbb{N}$	the number of idle periods in a layout l for a teacher t in a day d.
Decision variables	
$y_{dl} \in \{0,1\}$	indicates whether teacher t follows a layout $l \in L$ in a day d.
$x_{rdl} \in \{0,1\}$	indicates whether a requirement $r \in R_t$ follows a layout $l \in L$ in a day d.
Auxiliary variables	
$\widehat{\pi}_r \in \mathbb{N}$	the number of unmet weekly double lessons for a requirement r.

4.2.1 Column generation approach for EF2

The proposed column generation procedure defines the master problem as the linear relaxation of program (28)–(33). The initial master program contains a reduced number of artificial columns, to which we assign high costs. New columns are generated via $|T|$ pricing subproblems $\mathcal{Q}_t$, which select the most attractive weekly schedules (columns) for teachers. At each iteration, each subproblem $\mathcal{Q}_t$ is solved to obtain the column that has the smallest reduced cost, for teacher t, based on the dual values of the associated master program solution. The new columns are inserted into the master program, which is solved again, and the process continues until there is no column with reduced negative cost.

A pricing subproblem $\mathcal{Q}_t$ is responsible for generating the columns (weekly schedules) for a teacher $t \in T$. With the notation defined in Table 5, a mixed-integer program for $\mathcal{Q}_t$ can be written as:

$$\textbf{Minimize} \quad \overline{c}_t = \sum_{r\in R_t}\sum_{d\in D}\sum_{l\in L}(\alpha_5\widetilde{e}_{rl} + \alpha_6\widetilde{h}_l)x_{rdl} + \alpha_7\sum_{r\in R_t}\widehat{\pi}_r + \sum_{d\in D}\sum_{l\in L}(\alpha_8\widetilde{i}_{tdl} + \alpha_9\widetilde{\phi}_l)y_{dl}$$
$$- \xi^1_t - \sum_{c\in C_t}\sum_{d\in D}\sum_{h\in H}\xi^2_{cdh}\sum_{r\in R_{ct}}\sum_{l\in L}\widetilde{\mu}_{lh}x_{rdl} + \xi^3_t\sum_{d\in D}\sum_{l\in L}\widetilde{\phi}_l y_{dl} \tag{34}$$

Subject to:

$$\sum_{d\in D}\sum_{l\in L}\widetilde{a}_l x_{rdl} = \widetilde{\theta}_r \qquad \forall\ r \in R_t \tag{35}$$

$$\sum_{r\in R_{ct}}\sum_{l\in L}\widetilde{\mu}_{lh} x_{rdl} = 1 \qquad \forall\ c \in C_t;\ d \in D;\ h \in H \tag{36}$$

$$\sum_{l\in L} x_{rdl} = 1 \qquad \forall\ r \in R_t;\ d \in D \tag{37}$$

$$\sum_{l\in L} y_{dl} = 1 \qquad \forall\ d \in D \tag{38}$$

$$\sum_{r\in R_t}\sum_{l\in L}\widetilde{\mu}_{lh} x_{rdl} = \sum_{l\in L}\widetilde{\mu}_{lh} y_{dl} \qquad \forall\ d \in D;\ h \in H \tag{39}$$

$$\widehat{\pi}_r \geq \widetilde{\pi}_r - \sum_{d\in D}\sum_{l\in L}\widetilde{\omega}_l x_{rdl} \qquad \forall\ r \in R_t \tag{40}$$

$$\sum_{d\in D}\sum_{l\in L}\widetilde{\phi}_l y_{dl} \geq \widetilde{d}_t \tag{41}$$

$$\widehat{\pi}_r \geq 0 \qquad \forall\ r \in R_t \tag{42}$$

$$x_{rdl} \in \{0,1\} \qquad \forall\ r \in R_t;\ d \in D;\ l \in L \tag{43}$$

$$y_{dl} \in \{0,1\} \qquad \forall\ d \in D;\ l \in L \tag{44}$$

Objective function (34) is the reduced cost $\overline{c}_t$ based on the dual values ξ^1_t, ξ^2_{cdh} and ξ^3_t associated with constraints (29)–(31) of the master problem. Constraints (35) ensure that the required number of weekly lessons is scheduled for each requirement of a teacher t. Constraints (36) avoid clashes in classes' schedules. These constraints are necessary since a same teacher can be associated with different requirements (e.g., a teacher who teaches mathematics and physics to the same class). Constraints (37) and (38) ensure that a teacher t and their requirements $r \in R_t$ follow only one layout per day. The linking constraints (39) ensure that the daily schedules of a teacher t match his/her requirements' schedules. Constraints (40) compute the number of unmet double lessons for requirements. Finally, constraints (41) are valid inequalities stating that a teacher t does not work less than the minimum necessary number of days $\widetilde{d}_t$.

As before, to avoid violating teachers' unavailable periods, we preprocess the variables as follows.

$$y_{dl} = 0 \qquad \forall\ d \in D;\ l \in L;\ h \in H \setminus H_{td};\ \widetilde{\mu}_{lh} > 0 \tag{45}$$

$$x_{rdl} = 0 \qquad \forall\ r \in R_t;\ d \in D;\ l \in L;\ h \in H \setminus H_{td};\ \widetilde{\mu}_{lh} > 0 \tag{46}$$

5 A new parallel solution framework

We propose a parallel metaheuristic for the HSTP defined in Section 3 and modeled in Section 4. The approach builds on the *diversification-intensification memory based* (DIMB) framework proposed in Saviniec et al. (2018) for the same problem. The DIMB relies on a number of metaheuristic agents that work on parallel and share information via buffers of solutions. The method also uses intensification and diversification memories to improve the efficiency of the search.

The proposed *diversification-intensification memory based with column generation* (DIMB-CG) extends the DIMB by incorporating new agents based on the column geration proposed for model EF2, described in Section 4.2.1. The column generation based agents are

responsible for obtaining lower bounds for the problem. They also exploit the generated columns to build new feasible solutions that are used to diversify the search. Table 6 lists the main elements of the framework while Figure 2 shows their main interactions.

Table 6: DIMB-CG main elements.

Element	**Description**
Manager	Coordinates the exchange of information between the intensifiers and the diversification and intensification memories.
Intensifiers	Improvement metaheuristic methods used to improve incumbent solutions.
Diversifier	Heuristic method used to find solutions with different structures.
Memories	Pools of feasible solutions.
CG and CGCH	Column generation procedure to generate lower bounds and associated factibilization heuristic.

An overview of the whole algorithm is in the following. The manager procedure implements policies to guide the search and to update the diversification and intensification memories of solutions. It works by sending solutions (selected from one of the two memories) to intensifiers and receiving solutions from them, via the two intermediate buffers (the input and output buffers). The intensifiers explore the neighborhoods of the received solutions, during a pre-specified amount of time. The best solutions found during the search are sent back to the manager.

The agent CG is an implementation of the column generation approach on model EF2 described earlier. At each iteration of the column generation, a lower bound on the objective function value is obtained. Moreover, the dual values from the relaxed master problem are sent to a constructive heuristic, CGCH. The CGCH uses pricing subproblems created with this dual information to create new columns and complete solutions. The columns are sent back to the CG master problem while the complete solutions are sent to the diversification memory. The columns in the master are also sent to the diversifier agent through a pool of columns. This agent generates additional solutions to the diversification memory. In the following section, details on each of these agents are presented.

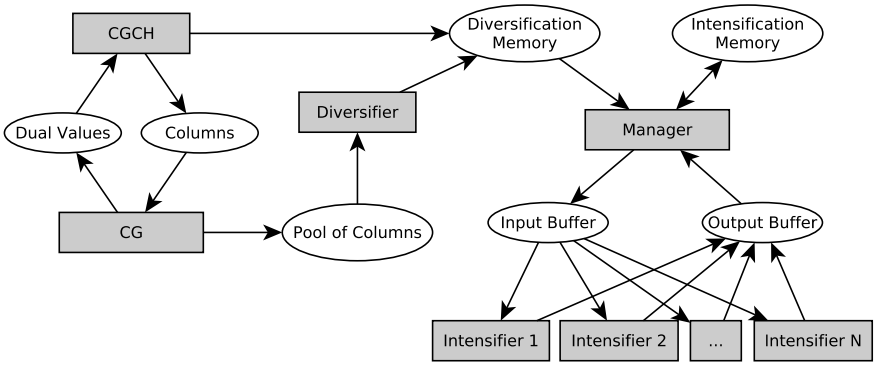


Figure 2: Interactions among elements of the DIMB-CG framework.

5.1 Implementation details

In this subsection, we detail the behavior of each of the agents listed earlier. We focus on the added elements with respect to Saviniec et al. (2018).

5.1.1 Manager and Intensifiers

Both the manager and intensifiers are implemented as in Saviniec et al. (2018). The manager controls the exchange of solutions between the intersifiers and the (diversification and intensification) memories. All intensifiers and both memories are initialized with solutions generated by the diversifier and the CGCH. Two events can trigger actions from the manager:

- An intensifier finds a new solution: this solution is compared with a random solution from the intensification memory. If the objective value of this new solution is better, it replaces the old solution in the intensification memory.
- An intensifier becomes idle: the manager feeds the intensifier agent with a solution from the diversification memory (with probability ρ) or from the intensification memory (with probability $1 - \rho$).

The intensifiers are copies of the Iterated Local Search algorithm used in Saviniec et al. (2018). A brief explanation of the algorithm has been added for convenience in Appendix B.

5.1.2 Column generation

The column generation agent (CG) works as described in Section 4.2.1 with the sole additional feature that the master problem is updated not only with columns generated from its own pricing problems $\mathcal{Q}_t$ (implemented as the MIP described in (34)–(44)) but also with columns generated by the CGCH.

5.1.3 Column generation constructive heuristic

For the CGCH agent, we propose a novel fix-and-optimize heuristic modified with a proximity search (Fischetti and Monaci, 2014) correction procedure. The solution is constructed by solving the pricing problems $\mathcal{Q}_t$ sequentially. After a given pricing problem is solved and a schedule is obtained for a given teacher t, this part of the solution is fixed. Subsequent subproblems include additional constraints to avoid any clashes with the already-fixed variables. Due to the myopic nature of the procedure, it frequently happens that the pricing problems become infeasible after enough variables are fixed.

Whenever an infeasible subproblem is found, we run a factibilization procedure based on proximity search. This is done by using a modified version of the model, where the objective function is to minimize the hamming distance between the new feasible solution and the current infeasible partial solution obtained so far. We use the compact model CF1 described in Appendix A, which relies on variables x_{rdh} indicating if a requirement r is fulfilled in period h of day d. Let J be the subset of variables x_{rdh}, in model CF1, which are already fixed in the current partial solution and $J_0 \subset J$ ($J_1 \subset J$) be the subset of variables that have values set to zero (one), the factibilization correction model can be written as:

$$\textbf{Minimize} \quad \sum_{x_{rdh} \in J_0} x_{rdh} + \sum_{x_{rdh} \in J_1} (1 - x_{rdh}) \tag{47}$$

Subject to:

$$(50) - (69)$$

$$x_{rdh} \in \{0, 1\} \qquad \forall\ r \in R;\ d \in D;\ h \in H \tag{48}$$

The objective function forces the problem to find the closest feasible solution to the current partial solution (in terms of the Hamming distance). This somehow preserves the structure of the solution built so far. Also, due to the strong relaxation grip of the model (Fischetti and Monaci, 2014), this factibilization problem tends to be easily solved.

5.1.4 Diversifier

The diversifier agent is responsible for feeding the diversification memory with solutions constructed by combining columns from the pool of columns. At each iteration, a random column is select for each teacher and concatenated to form a possible infeasible solution (with clashes in classes' schedules). The new solution is repaired and improved by an iterated local search algorithm (see Appendix B) during a pre-specified amount of time and sent to the diversification memory.

6 Computational experiments

In this section, we conduct several experiments to evaluate our models and the proposed parallel algorithm. Experiments concerning the models are discussed in Section 6.1 while those of the parallel algorithm are analyzed in Section 6.2. In both cases, the experiments use the problem described in Section 3 and two close variants. The three considered testbeds are:

- HSTP-A: the problem described in Section 3, with penalty parameters as suggested in Saviniec et al. (2018): $\alpha_2 = \alpha_3 = \alpha_4 = 100000$, $\alpha_5 = 100$, $\alpha_6 = 25$ and $\alpha_7 = \alpha_8 = \alpha_9 = \alpha_{10} = 10$.
- HSTP-B: the problem introduced by Souza (2000). In this problem, requisites 1 to 5 are considered to be hard, and requisites 7 to 9 are soft (requisites 6 and 10 are ignored). The objective function uses penalty parameters as suggested in Souza et al. (2004): $\alpha_2 = \alpha_3 = \alpha_4 = 100000$, $\alpha_5 = 10000$, $\alpha_7 = 1$, $\alpha_8 = 3$ and $\alpha_9 = 9$.
- HSTP-C: the problem HSTP-B with the inclusion of requisite 6 as a hard requisite. The objective function uses penalty parameters as suggested in Saviniec et al. (2018): $\alpha_2 = \alpha_3 = \alpha_4 = 100000$, $\alpha_5 = 10000$, $\alpha_6 = 5000$, $\alpha_7 = 1$, $\alpha_8 = 3$ and $\alpha_9 = 9$.

For all experiments with problems HSTP-A and HSTP-C, we use the instances proposed by Saviniec and Constantino (2017), described in Table 7. For all experiments with problem HSTP-B, we use the instances of Souza (2000), described in Table 8. In these tables, the columns present the number of classes ($|C|$), teachers ($|T|$), days ($|D|$), periods per day ($|H|$), the total number of weekly lessons ($\sum_{r \in R} \widetilde{\theta}_r$) and the total number of required double lessons ($\sum_{r \in R} \widetilde{\pi}_r$) for each instance.

All implementations were coded in C++ and compiled with GNU Compiler Collection 4.4.7. Concert Technology Libraries from IBM CPLEX 12.6 are used to solve the mixed-integer programming models. The experiments were performed on a computer Intel Xeon E5-2680v2 (2.8 GHz) with two cores and 128 GB of RAM, running *Red Hat Enterprise Linux 6.5*.

6.1 Experiments with the proposed models

In this section, the goal is to analyse the efficacy of the proposed extended formulations and column generation procedure (for EF2). We report the following experiments:

Table 7: Features of the instances proposed by Saviniec and Constantino (2017).

ID	Instance	$\|C\|$	$\|T\|$	$\|D\|$	$\|H\|$	$\sum_{r\in R}\widetilde{\theta}_r$	$\sum_{r\in R}\widetilde{\pi}_r$
1	CL-CEASD-2008-V-A	12	27	5	5	300	132
2	CL-CEASD-2008-V-B	12	27	5	5	300	132
3	CL-CECL-2011-M-A	13	31	5	5	325	144
4	CL-CECL-2011-M-B	13	31	5	5	325	143
5	CL-CECL-2011-N-A	9	28	5	5	225	107
6	CL-CECL-2011-V-A	14	29	5	5	350	164
7	CM-CECM-2011-M	20	51	5	5	500	234
8	CM-CECM-2011-N	8	30	5	5	200	96
9	CM-CECM-2011-V	13	34	5	5	325	142
10	CM-CEDB-2010-N	5	17	5	5	125	60
11	CM-CEUP-2008-V	16	35	5	5	400	192
12	CM-CEUP-2011-M	16	38	5	5	400	192
13	CM-CEUP-2011-N	3	15	5	5	75	36
14	CM-CEUP-2011-V	16	34	5	5	400	169
15	FA-EEF-2011-M	4	12	5	5	100	42
16	JNS-CEDPII-2011-M	8	19	5	5	200	85
17	JNS-CEDPII-2011-V	7	21	5	5	175	73
18	JNS-CEJXXIII-2011-M	5	18	5	5	125	60
19	JNS-CEJXXIII-2011-N	4	15	5	5	100	48
20	JNS-CEJXXIII-2011-V	5	18	5	5	125	60
21	MGA-CEDC-2011-M	19	37	5	5	475	210
22	MGA-CEDC-2011-V	12	31	5	5	300	131
23	MGA-CEGV-2011-M	31	62	5	5	775	352
24	MGA-CEGV-2011-V	32	75	5	5	800	357
25	MGA-CEJXXIII-2010-V	16	35	5	5	400	192
26	MGA-CEVB-2011-M	10	21	5	5	250	108
27	MGA-CEVB-2011-V	9	20	5	5	225	97
28	NE-CESVP-2011-M-A	18	45	5	5	450	212
29	NE-CESVP-2011-M-B	18	44	5	5	450	212
30	NE-CESVP-2011-M-C	18	45	5	5	450	211
31	NE-CESVP-2011-M-D	18	45	5	5	450	211
32	NE-CESVP-2011-V-A	16	44	5	5	400	183
33	NE-CESVP-2011-V-B	16	43	5	5	400	184
34	NE-CESVP-2011-V-C	16	43	5	5	400	182

Table 8: Features of the instances proposed by Souza (2000).

ID	$\|C\|$	$\|T\|$	$\|D\|$	$\|H\|$	$\sum_{r\in R}\widetilde{\theta}_r$	$\sum_{r\in R}\widetilde{\pi}_r$
1	3	8	5	5	75	21
2	6	14	5	5	150	29
3	8	16	5	5	200	4
4	12	23	5	5	300	41
5	13	31	5	5	325	71
6	14	30	5	5	350	63
7	20	33	5	5	500	84

Section 6.1.1: we compare the linear relaxation of EF1, EF1-cuts (EF1 augmented with Fenchel cuts), and EF2 with that of the compact formulation CF1, from Saviniec et al. (2018), on problem HSTP-A.

Section 6.1.2: we compare the performance of CPLEX when applied to EF1-cuts and CF1, in terms of the quality of primal and dual bounds and the number of instances with a proved optimal solution in a time limit of three hours.

Section 6.1.3: the two proposed formulations are adjusted to problem HSTP-B, and

we compare their results with results of formulations previously published for the same problem.

Section 6.1.4: the two proposed formulations are adjusted and applied to solve the problem HSTP-C.

6.1.1 The linear program of EF1 and EF2 compared to CF1 on problem HSTP-A

Table 9 compares the quality of the proposed formulations, EF1 and EF2, with formulation CF1 on problem HSTP-A. Tests were run with CPLEX's default settings. For each formulation, we present the execution time (Time) and the linear program objective function (LP). We also present the same metrics, for formulation EF1 with the addition of the Fenchel cuts described in Section 4.1.1 (EF1-cuts). The column Improvement provides, for each proposed formulation, the percentual improvement between its LP objective value and that of formulation CF1, which are calculated as $improvement = 100 \cdot (LP - LP_{CF1}) \div LP$.

The figures in Table 9 show that EF1 runs faster and obtains linear bounds 21.09 % better than CF1, in average. The addition of Fenchel cuts can slightly improve linear relaxation bounds with the price of a considerable increase in computational time. In the case of formulation EF2, the improvements obtained are even better than the ones obtained by EF1 and EF1-cuts. However, it required about three times more computational time than EF1-cuts.

Table 9: The linear relaxation of formulations EF1 and EF2 compared to CF1 on problem HSTP-A.

	CF1		EF1			EF1-cuts			EF2			
ID	Time (s)	LP	Time (s)	LP	Improvement	Time (s)	LP	Improvement	Time (s)	NoC	LB	Improvement
1	0.90	720	0.35	885	18.64	13.64	930	22.58	47.16	1782	950	24.21
2	0.85	720	0.32	885	18.64	10.37	930	22.58	47.89	1770	950	24.21
3	1.55	760	0.49	925	17.84	39.45	990	23.23	62.40	2032	1010	24.75
4	1.67	770	0.50	925	16.76	31.01	970	20.62	64.63	2065	990	22.22
5	0.90	623.34	0.30	788.34	20.93	23.01	791.67	21.26	35.82	1412	870	28.35
6	1.70	830	0.51	1075	22.79	48.33	1080	23.15	75.69	2183	1080	23.15
7	1.96	1275	0.55	1640	22.26	22.57	1680	24.11	118.60	3517	1690	24.56
8	0.27	707.06	0.16	864.38	18.20	1.28	910.77	22.37	22.19	1266	930	23.97
9	0.66	853.42	0.27	1065.42	19.90	3.91	1100	22.42	92.47	2703	1100	22.42
10	0.37	320	0.17	420	23.81	7.85	420	23.81	19.66	891	420	23.81
11	1.32	1040	0.41	1377.50	24.50	14.48	1391.67	25.27	108.70	3191	1400	25.71
12	1.12	1043.67	0.36	1401.12	25.51	7.09	1447	27.87	88.33	2971	1450	28.02
13	0.03	310	0.04	392.50	21.02	0.12	410	24.39	3.38	347	417.15	25.69
14	0.84	987.34	0.30	1193	17.24	7.98	1260	21.64	71.02	2767	1270	22.26
15	0.13	250	0.10	340	26.47	0.99	375	33.33	8.13	589	387	35.40
16	0.48	509	0.20	606.67	16.10	12.45	626.67	18.78	27.70	1195	640	20.47
17	0.42	470	0.19	525	10.48	10.62	540	12.96	21.11	999	580	18.97
18	0.38	325	0.17	425	23.53	7.66	445	26.97	19.87	869	455	28.57
19	0.33	270	0.14	350	22.86	2.97	360	25.00	13.01	673	360	25.00
20	0.37	306.67	0.17	422.50	27.42	5.88	446.67	31.34	20.35	933	470	34.75
21	1.77	1120	0.54	1405	20.28	30.27	1450	22.76	97.66	2933	1450	22.76
22	0.61	754	0.23	1009.09	25.28	4.60	1029.29	26.75	51.53	1915	1029.29	26.75
23	5.11	1929.53	1.21	2478.34	22.14	46.72	2500	22.82	302.46	6096	2505	22.97
24	4.13	2062.53	0.98	2597.23	20.59	45.53	2680	23.04	208.42	5741	2680	23.04
25	1.16	969	0.42	1295	25.17	34.10	1305	25.75	72.36	2546	1313.34	26.22
26	0.58	620	0.25	740	16.22	6.71	740	16.22	49.56	1686	750	17.33
27	0.45	584	0.19	710	17.75	4.28	730	20.00	37.01	1409	730	20.00
28	1.71	1110	0.59	1440	22.92	66.74	1450	23.45	119.56	3282	1570	29.30
29	1.58	1110	0.57	1440	22.92	57.28	1450	23.45	106.92	3128	1570	29.30
30	1.65	1094	0.56	1430	23.50	32.52	1450	24.55	126.22	3277	1590	31.19
31	1.44	1090	0.51	1410	22.70	18.24	1420	23.24	94.32	3075	1570	30.57
32	1.50	1010	0.55	1280	21.09	34.39	1295	22.01	72.45	2601	1390	27.34
33	1.32	1000	0.51	1275	21.57	35.88	1300	23.08	75.38	2696	1410	29.08
34	1.19	1016	0.43	1271.67	20.11	33.65	1305	22.15	61.65	2472	1390	26.91
Avg.:	1.19		0.39		21.09	21.25		23.32	71.87			25.57

6.1.2 The integer program of EF1-cuts compared to CF1 on problem HSTP-A

In this section, we compare both primal and dual bounds obtained by CPLEX when using the proposed formulation EF1-cuts and the compact formulation CF1. Table 10 presents, for each formulation, the execution time (Time), the best lower bound (LB), the objective value of the best integer solution (UB), and the associated optimality gap ($Gap = 100 \cdot (UB - LB) \div UB$) on the tested instances. In column Time, TL means that the time limit of three hours was reached without a proof of optimality. The best LB and UB found for each instance are shown in bold font. Finally, the last row (Rank) presents the number of instances in which optimality was proved or the best LB/UB were obtained.

Table 10: The integer program of formulation EF1-cuts compared to CF1 on problem HSTP-A (TL = 3 hours).

	CF1				EF1-cuts			
ID	Time (s)	LB	UB	Gap	Time (s)	LB	UB	Gap
1	TL	826.24	1290	35.95	TL	**940**	**1040**	9.62
2	TL	814.61	1260	35.35	TL	**941.43**	**950**	0.90
3	TL	839.17	1350	37.84	TL	**1000**	**1040**	3.85
4	TL	842.50	**1430**	41.08	TL	**990**	1900	47.89
5	TL	820	1020	19.61	1054.40	**870**	**870**	0.00
6	TL	880.36	1540	42.83	TL	**1080**	**1340**	19.40
7	TL	1647.82	**2325**	29.13	TL	**1690**	3525	52.06
8	2258.93	**950**	**950**	0.00	77.37	**950**	**950**	0.00
9	TL	1062.59	1430	25.69	83.30	**1100**	**1100**	0.00
10	4466.99	**420**	**420**	0.00	91.28	**420**	**420**	0.00
11	TL	1373.34	1885	27.14	323.13	**1400**	**1400**	0.00
12	TL	1422.16	2010	29.25	422.49	**1450**	**1450**	0.00
13	2.59	**420**	**420**	0.00	0.50	**420**	**420**	0.00
14	TL	1227.25	1650	25.62	529.50	**1270**	**1270**	0.00
15	1566.40	**395**	**395**	0.00	9.39	**395**	**395**	0.00
16	TL	579.17	750	22.78	517.54	**640**	**640**	0.00
17	TL	531.67	660	19.44	568.90	**580**	**580**	0.00
18	TL	442.50	490	9.69	120.26	**460**	**460**	0.00
19	TL	350	**360**	2.78	17.79	**360**	**360**	0.00
20	TL	417.34	520	19.74	2076.64	**470**	**470**	0.00
21	TL	1415	1990	28.89	5801.61	**1450**	**1450**	0.00
22	TL	1018.75	1245	18.17	295.88	**1035**	**1035**	0.00
23	TL	2458.45	**5715**	56.98	TL	**2504.28**	17790	85.92
24	TL	2604.80	**5115**	49.08	TL	**2680**	16825	84.07
25	TL	1301.67	1685	22.75	TL	**1315**	**1320**	0.38
26	TL	724.29	940	22.95	532.73	**750**	**750**	0.00
27	6105.45	**730**	**730**	0.00	204.87	**730**	**730**	0.00
28	TL	1457	**2250**	35.24	TL	**1565**	13420	88.34
29	TL	1313.58	2160	39.19	TL	**1545**	**2040**	24.26
30	TL	1444.29	**2165**	33.29	TL	**1560.84**	3500	55.40
31	TL	1425.38	**2070**	31.14	TL	**1554.45**	3480	55.33
32	TL	1275.56	**1830**	30.30	TL	**1360**	2185	37.76
33	TL	1273.17	**1920**	33.69	TL	**1390**	2755	49.55
34	TL	1269.45	**1920**	33.88	TL	**1380**	2205	37.41
Rank:		5	16	5		34	24	18
Avg.:				25.28				19.18

The results in the Table 10 indicate that formulation EF1-cuts can be used to obtain better bounds and prove optimality in more cases than the compact formulation CF1, in average.

6.1.3 Results of EF1-cuts and EF2 compared to state-of-the-art models on problem HSTP-B

In this section, we report the results obtained with formulations EF1 and EF2 when they are adapted to deal with problem HSTP-B. We use the instances shown in Table 8, which have optimal solutions known. Table 11 shows that the column generation procedure presented for EF2 is able to find optimal lower bounds for all instances tested. The table presents the execution time (Time), the number of generated columns (NoC), and the lower bound (LB) obtained for each instance. These results are competitive with Santos et al. (2012) and Dorneles et al. (2017), which reported the same lower bounds with their column generation approaches. For these tests, CPLEX was run with a single thread, and its remaining parameters were set to default.

Table 11: The lower bounds of formulation EF2 on problem HSTP-B.

ID	Optimal	Time (s)	NoC	LB	Gap
1	202	14.66	371	202	0.00
2	333	24.60	848	333	0.00
3	423	22.54	904	423	0.00
4	652	35.08	1364	652	0.00
5	762	39.73	1677	762	0.00
6	756	89.80	2025	756	0.00
7	1017	82.39	2216	1017	0.00

We also compare the performance of EF1-cuts with the compact formulation based on multi-commodity flow problems (CF-MCFP) by Dorneles et al. (2017). Table 12 shows, for both formulations, the best lower bounds obtained (LB), the best integer solutions (UB), and their optimality gaps (Gap) with a time limit of 2h (as used in Dorneles et al. (2017)). For these instances, we also report the linear programming objective functions for EF1 (column LP) and EF1-cuts (LP_{cuts}), and the time needed to obtain them.

Table 12: Formulation EF1-cuts compared to CF-MCFP on problem HSTP-B (TL = 2 hours).

	EF1		EF1-cuts					CF-MCFP		
ID	Time (s)	LP	Time (s)	LP_{cuts}	LB	UB	Gap	LB*	UB	Gap
1	0.08	189	1.66	190	**202**	**202**	0.00	190.24	**202**	5.82
2	0.67	333	18.84	333	**333**	**333**	0.00	**333**	**333**	0.00
3	0.39	414	19.29	414	**418**	444	5.86	413.99	**429**	3.50
4	2.08	643	83.92	652	**652**	**652**	0.00	**652**	**652**	0.00
5	4.05	756	273.95	757	**757.50**	**763**	0.72	756.02	777	2.70
6	6.06	738	375.48	750	**750**	935	19.79	739.52	**804**	8.02
7	12.69	999	842.92	1017	**1017**	–	–	1006.58	**1645**	38.81
Rank:					7	4	3	2	6	2

* Dorneles et al. (2017) did not report the lower bounds (LB) of their integer programs after two hours. We calculated these values by the formula $LB = UB \cdot (1 - Gap \div 100)$, where Gap was reported to be the CPLEX's optimality gap.

Formulations EF1 and CF-MCFP presented the same linear programming relaxation values shown in column LP. Formulation EF1-Cuts also improved these values via the Fenchel cuts. Regarding the integer programs, our formulation only performed worse than CF-MCFP in the number of best integer solutions. It ranked four best solutions against six of CF-MCFP. Our formulation also did not find a feasible solution to instance 7, during two hours. On the other hand, EF1-Cuts provided the best lower bounds for all instances and proved the optimality of three instances while CF-MCFP proved the

optimality only for two instances. We also observe that in instances 4 and 7, the Fenchel cuts led to optimal lower bounds.

6.1.4 Results of EF1-cuts and EF2 on problem HSTP-C

In this section, we report the results of formulations EF1-cuts and EF2 adapted to solve the problem HSTP-C. For this problem, we consider the instances of Table 7.

The results are shown in Table 13. For this problem, instances 15 and 22 are infeasible. For the remaining instances, the table compares the lower bounds of EF1-cuts and EF2. The columns labeled with (Gap) present, for each formulation, the gaps between its lower bounds (LB) and the best integer solutions (UB) of EF1-cuts. The best gaps are shown in bold font. For this problem, both formulations ranked the same number of best lower bounds. However, formulation EF1-cuts performed slightly better than EF2 in terms of the quality of obtained lower bounds. As an overall result, we could prove the optimality for 25 out of the 32 instances that are not infeasible.

Table 13: Results of formulations EF1-cuts and EF2 on problem HSTP-C.

	EF1		EF1-cuts (TL = 3 h)						EF2			
ID	Time (s)	LP	Time (s)	LP_{cuts}	Time (s)	LB	UB	Gap	Time (s)	NoC	LB	Gap
1	0.37	682.50	7.50	689	1446.54	699	699	**0.00**	49.06	1828	699	**0.00**
2	0.35	682.50	11.75	689	1689.78	699	699	**0.00**	49.83	1817	699	**0.00**
3	0.54	727.50	11.70	735	3900.23	742	742	**0.00**	58.25	2114	742	**0.00**
4	0.51	726.50	22.04	731	3977.16	734	734	**0.00**	59.99	2131	734	**0.00**
5	0.35	627	39.46	627	729.99	631	631	**0.00**	32.32	1427	631	**0.00**
6	0.56	771.50	24.53	772	2942.94	772	772	**0.00**	71.08	2282	772	**0.00**
7	0.58	1197.50	14.15	1209	10799.32	1212	1212	**0.00**	95.15	3281	1212	**0.00**
8	0.17	637.86	1.09	664.80	66.33	675	675	**0.00**	21.86	1293	669	0.89
9	0.35	797.18	6.84	805.25	247.16	807	807	**0.00**	59.83	2200	806.25	0.09
10	0.18	298	3.70	298	72.79	298	298	**0.00**	18.17	884	298	**0.00**
11	0.45	942.19	9.01	948.50	1972.01	961	961	**0.00**	76.99	2908	959.87	0.12
12	0.37	996.13	5.01	1014.74	498.25	1018	1018	**0.00**	82.87	2946	1014.74	0.32
13	0.04	261.50	0.05	265	0.72	273	273	**0.00**	2.51	302	270.67	0.85
14	0.36	911.23	8.33	930	761.33	933	933	**0.00**	48.35	2476	933	**0.00**
15					Infeasible							
16	0.24	472.67	5.39	477	485.21	481	481	**0.00**	22.55	1196	481	**0.00**
17	0.22	455.50	10.20	457	308.88	457	457	**0.00**	20.56	1078	457	**0.00**
18	0.18	309.25	3.81	316	100.77	319	319	**0.00**	19.62	859	319	**0.00**
19	0.15	251	3.09	254	14.75	254	254	**0.00**	15.60	738	254	**0.00**
20	0.19	310.50	3.79	317	224.26	325	325	**0.00**	24.83	1034	325	**0.00**
21	0.62	1036	13.41	1050	6163.85	1050	1050	**0.00**	73.70	2760	1050	**0.00**
22					Infeasible							
23	1.51	1789.19	39.77	1812	TL	1813	2357	**23.08**	273.51	6107	1813	**23.08**
24	1.03	1932.79	30.19	1983.50	TL	1988	2535	**21.58**	201.99	5832	1988	**21.58**
25	0.47	904	27.97	909	343.97	912	912	**0.00**	56.54	2408	912	**0.00**
26	0.29	570	8.94	570	245.98	570	570	**0.00**	37.02	1594	570	**0.00**
27	0.21	549.75	4.55	552	140.12	552	552	**0.00**	27.67	1322	552	**0.00**
28	0.63	1104	64.61	1105	TL	1122.13	1190	5.70	95.61	3067	1124	**5.55**
29	0.68	1095	43.33	1096	TL	1112.94	1306	14.78	115.67	3136	1116	**14.55**
30	0.61	1113	86.22	1114	TL	1131.40	1703	33.56	107.54	3154	1133	**33.47**
31	0.56	1112	60.81	1113	2618.76	1127	1127	**0.00**	79.59	2966	1127	**0.00**
32	0.57	999	53.11	999	TL	1010	1740	41.95	78.22	2658	1014	**41.72**
33	0.54	999.50	51.71	1001	TL	1016	1081	6.01	67.91	2613	1018	**5.83**
34	0.53	980	37.11	981	9355.78	1005	1005	**0.00**	73.85	2557	1005	**0.00**
Rank:								27				27
Avg.:								4.58				4.63

6.2 Experiments with the proposed parallel algorithm

In this section, we present two experiment sets to assess our parallel algorithm DIMB-CG.

> Section 6.2.1: we test the effect of inserting the CGCH extra columns into the master program of the CG procedure. We also analyze the behavior of the two strategies used in our constructive heuristic CGCH and the quality of its solutions.
>
> Section 6.2.2: we compare our algorithm DIMB-CG with the best parallel algorithm proposed in Saviniec et al. (2018).

In all tests, the algorithm DIMB-CG was allowed to execute during a time limit of 625 seconds, and 25 independent trials were carried out for each instance.

6.2.1 Analysis of the heuristic CGCH

In Table 14, we analyze the CG run-time speed-ups and the percentage of additional columns (PoAC) generated, when the extra columns of the heuristic CGCH are also inserted into the master program of the column generation (strategy denoted as CG+CGCH). This table shows that the convergence of the CG procedure is accelerated when the extra columns are also added to its master program. We observe that large speed-ups of 1.54, 2.29 and 1.23 were obtained for problems HSTP-A, HSTP-B and HSTP-C, respectively, due to the use of the new agent. However, we cannot observe any apparent relation between the number of extra columns generated and the speed-ups obtained.

Table 14: CG run-time speed-ups and percentage of additional columns generated when employing the heuristic CGCH.

	HSTP-A	HSTP-B	HSTP-C
Avg. speed-up:	1.54	2.29	1.23
Avg. PoAC:	12.05	-20.21	8.96

Table 15: Effectiveness of the fix-and-optimize and the proximity search strategies in constructing feasible solutions.

	HSTP-A	HSTP-B	HSTP-C
Avg. PoS1:	1.7	1.5	1.5
Avg. PoPPS:	85.8	89.7	85.8
Avg. PoS2:	100.0	100.0	99.5

In Table 15, we analyze the effectiveness of the fix-and-optimize and proximity search strategies used by our CGCH heuristic in constructing feasible solutions. In the table, Avg. PoS1 is the fix-and-optimize average percentage of success in constructing feasible solutions (e.g., PoS1 = 1.7 means that in 100 trials, the fix-and-optimize strategy generates only 1.7 % of feasible solutions). In the second row, Avg. PoPPS shows the average percentage of pricing problems that the fix-and-optimize strategy solves without failing. Namely, before it reaches an infeasible pricing problem (e.g., PoPPS = 85.0 means that, on average, the fix-and-optimize strategy succeed in solving 85 % of the pricing problems without reaching an infeasibility). Finally, Avg. PoS2 is the proximity search average percentage of success in repairing the infeasible partial solutions received from the fix-and-optimize strategy (e.g., PoS2 = 100.0 means that the proximity search strategy can repair all infeasible solutions received). This table shows that although the fix-and-optimize strategy fails with high rates, the proximity search strategy is able to repair almost all

Table 16: Quality of the CGCH solutions.

	HSTP-A	HSTP-B	HSTP-C
Avg. Opt. Gap:	19.17	2.74	10.57

infeasible partial solutions generated by the fix-and-optimize strategy, making this hybrid approach very effective in practice.

In Table 16, we analyze the quality of the best solutions constructed by the heuristic CGCH. This table shows that the proposed heuristic is very effective indeed: it builds feasible solutions with an average optimality gap of 19.17 % for problem HSTP-A and 10.57 % for problem HSTP-C. For problem HSTP-B, the produced solutions are almost optimal, with an average gap of only 2.74 %. We also observed that the less constrained the problem, more the proximity search was effective in not only repairing, but also improving the partial fix-and-optimize solutions.

6.2.2 Analysis of the algorithm DIMB-CG

In this section, we compare the proposed algorithm DIMB-CG with the best version of the parallel algorithm of Saviniec et al. (2018), called DIMB-19I1D-8-0.05.

In Table 17, we present the results for problem HSTP-A. Column "Best LB" shows the best lower bounds reported in Section 6.1. For each algorithm, we provide its median (Median) and best (Best) solutions, and the associated gaps (Gap) compared to the best-known lower bounds. The best results are shown in bold font. This table shows that the proposed algorithm performs better than DIMB-19I1D-8-0.05, both in terms of median and best solutions. In column "ToBS", we also show the DIMB-CG median execution time required to stagnate in the last improved solution (i.e. the best solution). This column shows that the most difficult instances for the method are instances 23 and 24. In instance 23, for example, the algorithm takes approximately 491 seconds to find near-optimal solutions ($\approx$ 8.6 % of optimality gap). For some small instances, such as 10, 15 and 18 to 20, the algorithm only takes a few seconds to find the optimal solutions.

In Table 18, we present the results for problem HSTP-B. This table shows that both algorithms were able to find the optimal solutions for all instances in the median case. Column (ToBS) shows that the algorithm DIMB-CG is almost instantaneous to find optimal solutions for these instances, except for instance 3, in which it takes around 31 seconds to reach the optimal solutions.

Table 19 presents the results for problem HSTP-C. This table shows that the algorithm DIMB-CG performs better than the algorithm DIMB-19I1D-8-0.05 in median solutions, but it is less effective in the number of best solutions. As the problem HSTP-C is more constrained than the problem HSTP-A, the algorithm DIMB-CG takes longer to make the last improvement, see column "ToBS". However, for most of the tested instances, it is still able to reach near-optimal solutions expending running time much smaller than the specified time limit.

Table 17: Results of the algorithm DIMB-CG compared to DIMB-19I1D-8-0.05 on problem HSTP-A.

		DIMB-CG (TL = 625 s)					DIMB-19I1D-8-0.05 (TL = 625 s)			
ID	Best LB	ToBS (s)	Median	Gap	Best	Gap	Median	Gap	Best	Gap
1	950	306.90	**960**	1.04	**950**	0.00	970	2.06	**950**	0.00
2	950	349.62	**960**	1.04	**950**	0.00	970	2.06	**950**	0.00
3	1010	204.64	**1010**	0.00	**1010**	0.00	1020	0.98	**1010**	0.00
4	990	148.24	**990**	0.00	**990**	0.00	1000	1.00	**990**	0.00
5	870	29.34	**870**	0.00	**870**	0.00	**870**	0.00	**870**	0.00
6	1080	220.62	**1080**	0.00	**1080**	0.00	1100	1.82	**1080**	0.00
7	1690	455.17	**1830**	7.65	**1770**	4.52	1880	10.11	1810	6.63
8	950	38.78	1045	9.09	**980**	3.06	**1030**	7.77	**980**	3.06
9	1100	269.50	**1160**	5.17	**1130**	2.65	1170	5.98	**1130**	2.65
10	420	8.42	**420**	0.00	**420**	0.00	**420**	0.00	**420**	0.00
11	1400	378.52	**1520**	7.89	1470	4.76	1530	8.50	**1450**	3.45
12	1450	332.04	**1580**	8.23	**1500**	3.33	1600	9.38	1530	5.23
13	420	46.25	**420**	0.00	**420**	0.00	**420**	0.00	**420**	0.00
14	1270	313.59	**1340**	5.22	**1290**	1.55	1360	6.62	1300	2.31
15	395	11.92	**395**	0.00	**395**	0.00	**395**	0.00	**395**	0.00
16	640	77.04	**640**	0.00	**640**	0.00	650	1.54	**640**	0.00
17	580	147.30	**580**	0.00	**580**	0.00	590	1.69	**580**	0.00
18	460	6.45	**460**	0.00	**460**	0.00	**460**	0.00	**460**	0.00
19	360	4.29	**360**	0.00	**360**	0.00	**360**	0.00	**360**	0.00
20	470	8.45	**470**	0.00	**470**	0.00	**470**	0.00	**470**	0.00
21	1450	432.95	**1520**	4.61	**1480**	2.03	1550	6.45	**1480**	2.03
22	1035	251.71	**1075**	3.72	**1045**	0.96	1095	5.48	1055	1.90
23	2505	490.41	**2740**	8.58	**2670**	6.18	2790	10.22	**2670**	6.18
24	2680	477.62	**2920**	8.22	**2840**	5.63	2970	9.76	2860	6.29
25	1315	401.50	**1380**	4.71	**1340**	1.87	1400	6.07	**1340**	1.87
26	750	284.05	**770**	2.60	**750**	0.00	780	3.85	**750**	0.00
27	730	249.81	**740**	1.35	**730**	0.00	750	2.67	**730**	0.00
28	1570	319.46	**1600**	1.88	**1590**	1.26	1620	3.09	**1590**	1.26
29	1570	444.11	**1600**	1.88	**1580**	0.63	1620	3.09	1590	1.26
30	1590	343.04	**1620**	1.85	1610	1.24	1640	3.05	**1600**	0.63
31	1570	421.71	**1620**	3.09	1600	1.88	1640	4.27	**1590**	1.26
32	1390	437.21	**1420**	2.11	**1410**	1.42	1440	3.47	**1410**	1.42
33	1410	307.89	**1450**	2.76	1440	2.08	1460	3.42	**1430**	1.40
34	1390	427.91	**1430**	2.80	1420	2.11	1450	4.14	**1410**	1.42
Rank:			33		29		8		28	
Avg.:				2.81		1.39		3.78		1.48

Table 18: Results of the algorithm DIMB-CG compared to DIMB-19I1D-8-0.05 on problem HSTP-B.

		DIMB-CG (TL = 625 s)					DIMB-19I1D-8-0.05 (TL = 625 s)			
ID	Best LB	ToBS (s)	Median	Gap	Best	Gap	Median	Gap	Best	Gap
1	202	2.17	**202**	0.00	**202**	0.00	**202**	0.00	**202**	0.00
2	333	2.44	**333**	0.00	**333**	0.00	**333**	0.00	**333**	0.00
3	423	30.55	**423**	0.00	**423**	0.00	**423**	0.00	**423**	0.00
4	652	4.67	**652**	0.00	**652**	0.00	**652**	0.00	**652**	0.00
5	762	2.94	**762**	0.00	**762**	0.00	**762**	0.00	**762**	0.00
6	756	3.00	**756**	0.00	**756**	0.00	**756**	0.00	**756**	0.00
7	1017	3.62	**1017**	0.00	**1017**	0.00	**1017**	0.00	**1017**	0.00
Rank:			7		7		7		7	
Avg.:				0.00		0.00		0.00		0.00

7 Conclusions

In this paper, we conducted a study with mixed-integer programming formulations and parallel metaheuristic based algorithms for high school timetabling problems. We pro-

Table 19: Results of the algorithm DIMB-CG compared to DIMB-19I1D-8-0.05 on problem HSTP-C.

		DIMB-CG (TL = 625 s)					DIMB-19I1D-8-0.05 (TL = 625 s)			
ID	Best LB	ToBS (s)	Median	Gap	Best	Gap	Median	Gap	Best	Gap
1	699	364.74	**708**	1.27	701	0.29	710	1.55	**700**	0.14
2	699	409.47	**707**	1.13	**701**	0.29	711	1.69	**701**	0.29
3	742	389.30	**747**	0.67	**743**	0.13	749	0.93	**743**	0.13
4	734	419.58	**738**	0.54	**735**	0.14	743	1.21	**735**	0.14
5	631	93.26	**631**	0.00	**631**	0.00	**631**	0.00	**631**	0.00
6	772	403.62	**772**	0.00	**772**	0.00	777	0.64	**772**	0.00
7	1212	495.65	**1259**	3.73	**1237**	2.02	1270	4.57	1239	2.18
8	675	58.63	**697**	3.16	**679**	0.59	698	3.30	686	1.60
9	807	357.22	**821**	1.71	**812**	0.62	827	2.42	814	0.86
10	298	10.42	**298**	0.00	**298**	0.00	**298**	0.00	**298**	0.00
11	961	467.90	**998**	3.71	987	2.63	1005	4.38	**985**	2.44
12	1018	334.13	**1039**	2.02	1032	1.36	1044	2.49	**1030**	1.17
13	273	4.11	**273**	0.00	**273**	0.00	**273**	0.00	**273**	0.00
14	933	440.61	**946**	1.37	940	0.74	954	2.20	**939**	0.64
15					Infeasible					
16	481	173.54	**483**	0.41	**481**	0.00	484	0.62	**481**	0.00
17	457	309.41	**460**	0.65	**457**	0.00	461	0.87	**457**	0.00
18	319	22.51	**319**	0.00	**319**	0.00	**319**	0.00	**319**	0.00
19	254	2.32	**254**	0.00	**254**	0.00	**254**	0.00	**254**	0.00
20	325	18.41	**325**	0.00	**325**	0.00	**325**	0.00	**325**	0.00
21	1050	452.70	**1066**	1.50	1059	0.85	1074	2.23	**1058**	0.76
22					Infeasible					
23	1813	534.02	**1887**	3.92	1874	3.26	1910	5.08	**1867**	2.89
24	1988	565.76	**2059**	3.45	2042	2.64	2079	4.38	**2038**	2.45
25	912	485.79	**937**	2.67	**920**	0.87	944	3.39	926	1.51
26	570	219.81	**575**	0.87	571	0.18	576	1.04	**570**	0.00
27	552	306.29	**555**	0.54	**552**	0.00	556	0.72	**552**	0.00
28	1124	459.30	**1142**	1.58	1132	0.71	1150	2.26	**1127**	0.27
29	1116	511.92	**1137**	1.85	1130	1.24	1145	2.53	**1127**	0.98
30	1133	426.27	**1153**	1.73	**1142**	0.79	1164	2.66	**1142**	0.79
31	1127	511.13	**1152**	2.17	1140	1.14	1165	3.26	**1137**	0.88
32	1014	506.90	**1036**	2.12	1026	1.17	1043	2.78	**1024**	0.98
33	1018	352.11	**1039**	2.02	**1028**	0.97	1047	2.77	1034	1.55
34	1005	417.23	**1032**	2.62	1020	1.47	1041	3.46	**1015**	0.99
Rank:			32		19		6		27	
Avg.:				1.48		0.75		1.98		0.74

posed two extended pattern-based formulations, and a cooperative parallel algorithm.

The first proposed formulation, EF1, could be solved with black-box solvers and showed to have similar performance to the best available compact formulations. Also effective was the introduction of Fenchel cuts wich increased the obtained lower bounds with reasonable computational costs. The second formulation EF2 was solved within a column generation framework and also showed competitive results with the state-of-the-art extended formulations (Santos et al., 2012; Dorneles et al., 2017), often obtaining optimal lower bounds.

The proposed parallel algorithm DIMB-CG found near-optimal solutions for all studied problems outperforming previous methods in three variants of the HSTP. The main contribution of this algorithm is the introduction of new agents that exploit the developed column generation method for EF2 in order to generate new feasible solutions. This was done using a novel combination of fix-and-optimize and proximity search heuristics. The combination of these two heuristics proved successful and can certainly be extended to

other problems. Future work also includes the investigation of branch-and-price algorithms making use of the proposed column generation approach.

Acknowledgments

This research was supported by FAPESP-Brazil, Grants 2013/13563-3 and 2015/10032-2.

References

S. M. Al-Yakoob and H. D. Sherali. Mathematical models and algorithms for a high school timetabling problem. *Computers & Operations Research*, 61:56–68, 2015.

G. N. Beligiannis, C. N. Moschopoulos, G. P. Kaperonis, and S. D. Likothanassis. Applying evolutionary computation to the school timetabling problem: The greek case. *Computers & Operations Research*, 35(4):1265–1280, 2008.

G. N. Beligiannis, C. Moschopoulos, and S. D. Likothanassis. A genetic algorithm approach to school timetabling. *Journal of the Operational Research Society*, 60(1):23–42, 2009.

A. E. Boyd. Solving 0/1 integer programs with enumeration cutting planes. *Annals of Operations Research*, 50(1):61–72, 1994a.

E. A. Boyd. Fenchel cutting planes for integer programs. *Operations Research*, 42(1): 53–64, 1994b.

D. De Werra. An introduction to timetabling. *European Journal of Operational Research*, 19(2):151–162, 1985.

G. Desaulniers, J. Desrosiers, and M. M. Solomon. *Column generation*, volume 5. Springer Science & Business Media, 1 edition, 2005.

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

Á. P. Dorneles, O. C. de Araújo, and L. S. Buriol. A column generation approach to high school timetabling modeled as a multicommodity flow problem. *European Journal of Operational Research*, 256(3):685–695, 2017.

S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multi-commodity flow problems. In *16th Annual Symposium on Foundations of Computer Science*, pages 184–193. IEEE, 1975.

M. Fischetti and M. Monaci. Proximity search for 0-1 mixed-integer convex programming. *Journal of Heuristics*, 20(6):709–731, 2014.

G. H. Fonseca, H. G. Santos, and E. G. Carrano. Integrating matheuristics and metaheuristics for timetabling. *Computers & Operations Research*, 74:108–117, 2016a.

G. H. Fonseca, H. G. Santos, E. G. Carrano, and T. J. Stidsen. Integer programming techniques for educational timetabling. *European Journal of Operational Research*, 262 (1):28–39, 2017.

G. H. G. Fonseca, H. G. Santos, T. Â. M. Toffolo, S. S. Brito, and M. J. F. Souza. Goal solver: a hybrid local search based solver for high school timetabling. *Annals of Operations Research*, 239(1):77–97, 2016b.

C. C. Gotlieb. The construction of class-teacher timetables. In C. M. Popplewell, editor, *Proceedings of the IFIP Congress 62 in Information Processing 1963*, pages 73–77, 1963.

W. Junginger. Timetabling in germany – a survey. *Interfaces*, 16(4):66–74, 1986.

S. Kristiansen, M. Sørensen, and T. R. Stidsen. Integer programming for the generalized high school timetabling problem. *Journal of Scheduling*, 18(4):377–392, 2015.

R. Lewis. A survey of metaheuristic-based techniques for university timetabling problems. *OR Spectrum*, 30(1):167–190, 2008.

K. Papoutsis, C. Valouxis, and E. Housos. A column generation approach for the timetabling problem of greek high schools. *Journal of the Operational Research Society*, 54(3):230–238, 2003.

N. Pillay. A survey of school timetabling research. *Annals of Operations Research*, 218 (1):261–293, 2014.

G. Post, S. Ahmadi, S. Daskalaki, J. Kingston, J. Kyngas, C. Nurmi, and D. Ranson. An xml format for benchmarks in high school timetabling. *Annals of Operations Research*, 194(1):385–397, 2012.

G. Post, J. H. Kingston, S. Ahmadi, S. Daskalaki, C. Gogos, J. Kyngas, C. Nurmi, N. Musliu, N. Pillay, H. Santos, and A. Schaerf. Xhstt: an xml archive for high school timetabling problems in different countries. *Annals of Operations Research*, 218(1): 295–301, 2014.

R. Qu, E. K. Burke, B. McCollum, L. T. Merlot, and S. Y. Lee. A survey of search methodologies and automated system development for examination timetabling. *Journal of scheduling*, 12(1):55–89, 2009.

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

H. Santos, E. Uchoa, L. Ochi, and N. Maculan. Strong bounds with cut and column generation for class-teacher timetabling. *Annals of Operations Research*, 194(1):399–412, 2012.

L. Saviniec and A. A. Constantino. Effective local search algorithms for high school timetabling problems. *Applied Soft Computing*, 60:363–373, 2017.

L. Saviniec, M. O. Santos, and A. M. Costa. Parallel local search algorithms for high school timetabling problems. *European Journal of Operational Research*, 265(1):81–98, 2018.

V. I. Skoullis, I. X. Tassopoulos, and G. N. Beligiannis. Solving the high school timetabling problem using a hybrid cat swarm optimization based algorithm. *Applied Soft Computing*, 52:277–289, 2017.

M. J. F. Souza. *Programação de Horários em Escolas: Uma Aproximação por Meta-heurísticas*. PhD thesis, Programa de Pós-Graduação em Engenharia de Sistemas e Computação - Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2000.

M. J. F. Souza, N. Maculan, and L. S. Ochi. *A GRASP-Tabu Search Algorithm for Solving School Timetabling Problems*, pages 659–672. Springer US, 2004.

M. Sørensen and F. H. Dahms. A two-stage decomposition of high school timetabling applied to cases in denmark. *Computers & Operations Research*, 43:36–49, 2014.

I. X. Tassopoulos and G. N. Beligiannis. A hybrid particle swarm optimization based algorithm for high school timetabling problems. *Applied Soft Computing*, 12(11):3472–3489, 2012.

D. Zhang, Y. Liu, R. M'Hallah, and S. C. Leung. A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems. *European Journal of Operational Research*, 203(3):550–558, 2010.

Appendices

A Compact formulation

Table 20: Notation used in formulation CF1.

Notation	Definition
Decision variables	
$x_{rdh} \in \{0,1\}$	indicates whether a requirement $r \in R$ is assigned to a period $h \in H$ of a day $d \in D$.
Auxiliary variables	
$\widehat{\delta}_{rd} \in \mathbb{N}$	the number of lessons that exceeds the limit $\widetilde{\delta}_r$ in a day d for a requirement r.
$\widehat{j}_{rdh} \in \{0,1\}$	indicates whether the schedule of a requirement r has a gap at period h of a day d.
$\widehat{\phi}_{rdh} \in \{0,1\}$	indicates whether a requirement r has a double lesson at periods $h-1$ and h of a day d.
$\widehat{\pi}_r \in \mathbb{N}$	the number of unmet weekly double lessons for a requirement r.
$\widehat{d}_{td} \in \{0,1\}$	indicates whether a teacher t works in a day d.
$\widehat{i}_{tdk} \in \{0,1\}$	indicates whether a teacher t is idle at period k of a day d.
$\widehat{\beta} \in \mathbb{N}$	the greatest value among all teachers' extra working days.

This section describes a reformulation of the MIP model proposed in Saviniec et al. (2018), denoted here by CF1. In this reformulation, a timetable output can be represented by a set of binary variables x_{rdh}, for all $(r,d,h) \in R \times D \times H$. Let us consider the notation defined in Table 20, the HSTP can be formulated as follows:

$$\textbf{Minimize} \quad \alpha_5 \sum_{r\in R}\sum_{d\in D} \widehat{\delta}_{rd} + \alpha_6 \sum_{r\in R}\sum_{d\in D}\sum_{h=1}^{|H|-2} \widehat{j}_{rdh} + \alpha_7 \sum_{r\in R} \widehat{\pi}_r + \alpha_8 \sum_{t\in T}\sum_{d\in D}\sum_{k=1}^{|H_{td}|-2} \widehat{i}_{tdk} + \alpha_9 \sum_{t\in T}\sum_{d\in D} \widehat{d}_{td} + \alpha_{10} \cdot \widehat{\beta} \tag{49}$$

Subject to:

$$\sum_{d\in D}\sum_{h\in H} x_{rdh} = \widetilde{\theta}_r \qquad \forall\ r \in R \tag{50}$$

$$\sum_{r\in R_c} x_{rdh} = 1 \qquad \forall\ c \in C;\ d \in D;\ h \in H \tag{51}$$

$$\sum_{r\in R_t} x_{rdh} \leq 1 \qquad \forall\ t \in T;\ d \in D;\ h \in H_{td} \tag{52}$$

$$\sum_{r\in R_t} x_{rdh} = 0 \qquad \forall\ t \in T;\ d \in D;\ h \in H \setminus H_{td} \tag{53}$$

$$\widehat{\delta}_{rd} \geq \sum_{h\in H} x_{rdh} - \widetilde{\delta}_r \qquad \forall\ r \in R;\ d \in D \tag{54}$$

$$\widehat{j}_{rdh} \geq x_{rdi} - x_{rdh} + x_{rdj} - 1 \qquad \forall\ r \in R;\ d \in D;\ h = 1,\cdots,|H|-2;\ i = 0,\cdots,h-1;\ j = h+1,\cdots,|H|-1 \tag{55}$$

$$\widehat{\phi}_{rdh} \leq x_{rdh} \qquad \forall\ r \in R;\ d \in D;\ h = 1,\cdots,|H|-1 \tag{56}$$

$$\widehat{\phi}_{rdh} \leq x_{r,d,h-1} \qquad \forall\ r \in R;\ d \in D;\ h = 1,\cdots,|H|-1 \tag{57}$$

$$\widehat{\phi}_{rdh} \leq 1 - \widehat{\phi}_{r,d,h-1} \qquad \forall\ r \in R;\ d \in D;\ h = 2,\cdots,|H|-1 \tag{58}$$

$$\widehat{\pi}_r \geq \widetilde{\pi}_r - \sum_{d\in D}\sum_{h=1}^{|H|-1} \widehat{\phi}_{rdh} \qquad \forall\ r \in R \tag{59}$$

$$\widehat{i}_{tdk} \geq \sum_{r\in R_t}(x_{r,d,h_i} - x_{r,d,h_k} + x_{r,d,h_j}) - 1 \qquad \forall\ t \in T;\ d \in D;\ k = 1,\cdots,|H_{td}|-2;\ i = 0,\cdots,k-1;\ j = k+1,\cdots,|H_{td}|-1;\ h_i,h_k,h_j \in H_{td} \tag{60}$$

$$\widehat{d}_{td} \geq \sum_{r\in R_t} x_{rdh} \qquad \forall\ t \in T;\ d \in D;\ h \in H_{td} \tag{61}$$

$$\widehat{\beta} \geq \sum_{d\in D} \widehat{d}_{td} - \widetilde{d}_t \qquad \forall\ t \in T \tag{62}$$

$$\widehat{\delta}_{rd} \geq 0 \qquad \forall\ r \in R;\ d \in D \tag{63}$$

$$\widehat{j}_{rdh} \geq 0 \qquad \forall\ r \in R;\ d \in D;\ h = 1,\cdots,|H|-2 \tag{64}$$

$$\widehat{\phi}_{rdh} \geq 0 \qquad \forall\ r \in R;\ d \in D;\ h = 1,\cdots,|H|-1 \tag{65}$$

$$\widehat{\pi}_r \geq 0 \qquad \forall\ r \in R \tag{66}$$

$$\widehat{i}_{tdk} \geq 0 \qquad \forall\ t \in T;\ d \in D;\ k = 1,\cdots,|H_{td}|-2 \tag{67}$$

$$\widehat{d}_{td} \geq 0 \qquad \forall\ t \in T;\ d \in D \tag{68}$$

$$\widehat{\beta} \geq 0 \tag{69}$$

$$x_{rdh} \in \{0,1\} \qquad \forall\ r \in R;\ d \in D;\ h \in H \tag{70}$$

The objective function (49) minimizes the cost of breaking soft requisites 5 to 10. Constraints (50) to (53) model the hard requisites. Constraints (50) ensure that all required lessons are scheduled. Constraints (51) to (52) avoid clashes in classes and teachers' schedules, and constraints (53) prevent the assignment of teachers to unavailable periods.

The other constraints are related to the soft requisites. The requisite 5 is modeled by constraints (54), in which the auxiliary variables $\widehat{\delta}_{rd}$ measure the excess of lessons in

daily workloads that are greater than $\widetilde{\delta}_r$, for each requirement r. The number of gaps in requirements' schedules are computed with the introduction of the auxiliary variables $\widehat{j}_{rdh}$ and the constraints (55). The variable $\widehat{j}_{rdh}$ indicates whether a requirement r has a gap at period h of a day d. The requisite regarding the meeting of double lessons is modeled by the set of constraints (56) to (59). For each requirement r, the auxiliary variables $\widehat{\phi}_{rdh}$ identify whether a double lesson ends in a period $h \in \{1, \ldots, |H| - 1\}$ of a day $d \in D$, and the number of unmet weekly double lessons is quantified in constraint (59), with the help of the additional auxiliary variable $\widehat{\pi}_r$.

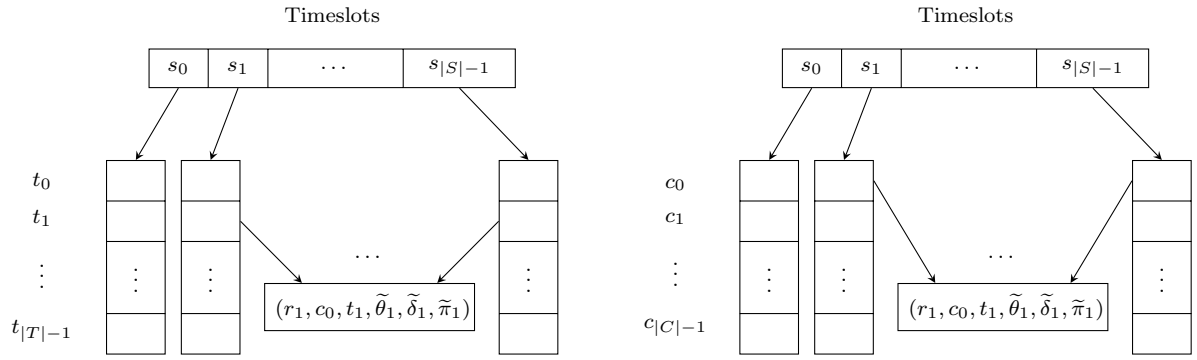
To compute idle periods in teachers' schedules, we introduce the constraints (60), and the auxiliary variables $\widehat{i}_{tdk}$, that flag if a teacher t is idle in a period $h_k \in H_{td}$ $(k = 1, \cdots, |H_{td}| - 2)$ of a day $d \in D$. The computation of idle periods in teachers' schedules is similar to the computation of gaps in requirements' schedules, except that the variables $\widehat{i}$ are omitted for periods in which the teachers are unavailable (Saviniec et al., 2018). As an example, consider $H = \{0, 1, 2, 3, 4\}$, and suppose that a teacher t is unavailable at periods 2 and 3 during a day d. Therefore, $H_{td} = \{0, 1, 4\}$ and only the variable $\widehat{i}_{t,d,1}$ is considered, which means that unavailable periods are disregarded. In constraints (61), we model the compactness of teachers' schedules. For each teacher t, the auxiliary variables $\widehat{d}_{td}$ flag if he/she is working in a day d. Finally, the last set of constraints (62) model the balancing on teachers' extra working days.

B The stand-alone iterated local search

This section describes the sequential Iterated Local Search (ILS) adapted from Saviniec et al. (2018), that we employ in our parallel algorithm of Section 5. We use two versions of this metaheuristic, one works within the diversifier agent while the other works within the intensifier agents. Each version also manipulates a different type of timetable encoding, that we call type I and type II. The timetable type I is manipulated by the ILS that works within the diversifier and the timetable type II is manipulated by the ILS that works within the intensifiers. Following, we discuss the two timetable encodings (Section B.1), the neighborhood structure and objective function evaluation (Section B.2), then lastly, the ILS pseudo-code is presented (Section B.3).

B.1 Timetable encodings

Let S be an array of all timeslots $(d, h) \in D \times H$, we illustrate the timetables type I and II in Figure 3. Both of them are based on the persistent data structure proposed in Saviniec et al. (2018). In both cases, the core of the encodings is an array of timeslots. In a timetabling encoding, either type I or II, each timeslot points to a second-layer array of pointers, each one of those pointing to a tuple $(r \in R, c \in C, t \in T, \widetilde{\theta}_r, \widetilde{\delta}_r, \widetilde{\pi}_r)$ that represents a requirement $r \in R$ and its parameters, the associated class, teacher, number of weekly lessons, limit of lessons per day, and number of required double lessons, respectively. In timetable encoding type I (resp. type II), the i^{th} position in the second-layer array indicates a requirement associated to the i^{th} teacher (resp. class). In the remainder of this paper, to improve presentation, instead of showing a pointer to a requirement tuple itself, we show only the teacher or class ID belonging to the tuple being pointed by the pointer.



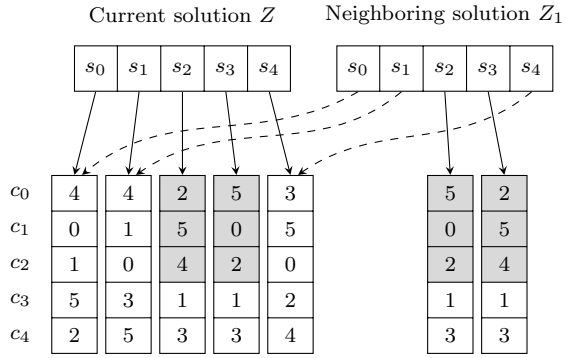
(a) Timetable type I: teachers view.

(b) Timetable type II: classes view.

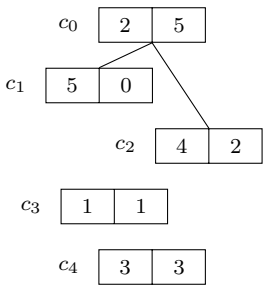
Figure 3: Timetable encodings.

B.2 Neighborhood structure and objective function

Neighboring solutions are generated via the torque neighborhood operator (TQ) (Saviniec and Constantino, 2017). This operator identifies chains of two-swap moves that must be performed together so that new clashes in classes' (or teachers') schedules are avoided. A *two-swap* move consists in exchanging the requirement tuples of a teacher $t \in T$ (or class $c \in C$), that are assigned to two different timeslots. Chains of two-swap moves are identified by conflict graphs. Figure 4 shows an example of a TQ move with the timetable type II. In this example, we identify all possible moves for tuples that are assigned to timeslots s_2 and s_3 of the timetable solution Z, shown in Figure 4a. The conflict graph is shown in Figure 4b, in which there is one vertex for each class, representing its tuples (remember, we are showing only the associated teachers), and edges connecting conflicting vertexes. The neighboring solution Z_1 (Figure 4a) is generated by applying a two-swap move for each class in the connected component $\{c_0, c_1, c_2\}$. Neighboring solutions with timetable type I are obtained similarly as made with timetable type II.



(a) Example of a TQ move applied to a timetable solution Z of type II.



(b) Conflict graph for tuples assigned to timeslots s_2 and s_3 in solution Z of Figure 4a.

Figure 4: Neighborhood structure with timetable type II.

Both ILS versions work in an infeasible space of solutions in which some of the hard requisites are relaxed and penalized in the objective function. For the ILS version which works inside the intensifier agents, the timetable encoding type II ensures that hard requisites 1 and 2 are never violated by TQ moves. Therefore, we relax hard requisites 3 and 4, and penalize them using the following objective function:

$$Minimize \quad f(Z) = (1) + \sum_{i \in \{3,4\}} \alpha_i \cdot V_i \tag{71}$$

Which is the objective function of the model EF1 plus the number of violations V_i, for hard requisites 3 and 4, times a penalty parameter α_i.

For the ILS version which works inside the diversifier agent, the timetabling encoding type I ensures that hard requisites 1 and 3 are never broken by the employed operator. Hence, we relax hard requisites 2 and 4, and penalize them according to the following objective function:

$$Minimize \quad f(X) = (1) + \sum_{i \in \{2,4\}} \alpha_i \cdot V_i \tag{72}$$

B.3 The ILS algorithm

Let SP be the array of all timeslot pairs (s_i, s_j), for $i, j = 0, \cdots, |S| - 1$ and $i \neq j$, we show the pseudo-code of the sequential ILS in Algorithm 1. At each iteration of the main loop (lines 3 to 23), the algorithm performs a random perturbation in its global best solution (line 4), followed by a local search in the resulting solution (lines 5 to 18), to reach a local optimum. If the local optimum is better than or equal to the global best solution (line 19), it is accepted. Otherwise, it is rejected. The algorithm is controlled by either, a maximum number of iterations or a time limit. The local search phase (lines 5 to 18) is a first improvement strategy that starts in a random index of SP (line 7), and inspects the next $|SP|$ consecutive timeslot pairs (lines 8 to 17) to find new improving solutions.

Algorithm 1 Pseudo-code of the sequential ILS adapted from Saviniec et al. (2018).

```
ILS(Z_0, timeOut, iterOut)
 1  Z* = Z_0
 2  k = 0
 3  while (elapsedTime < timeOut and k < iterOut) {
 4      Z = PERTURBATION(Z*)                    // a random TQ move
 5      do {
 6          best = f(Z)
 7          i = take a random integer in the range 0 ≤ i ≤ |SP| − 1
 8          for (j = 1 to |SP|) {
 9              Compute the connected components for tuples assigned to timeslots sp_i of Z
10              for (each connected component) {
11                  Generate a new neighboring solution Z' from Z
12                  if (f(Z') ≤ f(Z)) {
13                      Z = Z'
14                  }
15              }
16              i = (i + 1) mod |SP|
17          }
18      } while (f(Z) < best)
19      if (f(Z) ≤ f(Z*)) {
20          Z* = Z
21      }
22      k = k + 1
23  }
24  return Z*
```