A Parallel Multi-Start Iterated Local Search and a Proximity Relax-and-Fix heuristic for High School Timetabling Problem

Landir Saviniec*, Maristela Oliveira Santos*, Alysson Machado Costa†
landir.saviniec@gmail.com, mari@icmc.usp.br,
alysson.costa@unimelb.edu.au
*Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo – Brasil
†School of Mathematics and Statistics – The University of Melbourne – Australia

ABSTRACT

The high school timetabling problem has been frequently considered in the literature, but few studies employing parallel strategies have been proposed. In this paper, we propose a parallel cooperative multi-start approach. The method constructs an initial feasible solution via a Proximity Relax-and-Fix heuristic and applies a Parallel Multi-Start Iterated Local Search to further improve the initial solution. Computational experiments with practical instances show that our approach outperforms the state-of-the-art algorithms for variants of the problem. Our results clearly demonstrate the power of this type of approach to deal with complex combinatorial problems, such as the high school timetabling problem.

KEYWORDS. High School Timetabling Problem. Proximity Relax-and-Fix. Parallel Multi-Start Iterated Local Search.

Paper topics: Parallel Algorithms in CPU & GPU. Metaheuristics. Combinatorial Optimization.

Introduction

The scientific literature has branched educational timetabling problems in three main families: University course timetabling [Lewis et al., 2007; Di Gaspero et al., 2007], high school timetabling [Pillay, 2014] and examination timetabling [McCollum et al., 2007]. Each category contains a rich number of specific constraints and particularities.

In this paper, we investigate parallel metaheuristics for the high school timetabling problem (HSTP). Due to its combinatorial characteristics, the HSTP resolution has been mainly based on approximations by sequential metaheuristics. According to our knowledge, only four studies have addressed the design of parallel metaheuristics for HSTP. Abramson [1991] proposed a Simulated Annealing which analyzes several simultaneous moves. Abramson and Abela [1992] proposed a Genetic Algorithm that evaluates several individuals of the population in parallel. Srndic et al. [2009] proposed a Genetic Algorithm which splits the global population into small islands that are managed in parallel. Saviniec et al. [2015] proposed an Iterated Local Search that, at each iteration, chooses the best local minimum among a set of local minima returned by multiples local search procedures running simultaneously.

Alba et al. [2013] reviewed the literature of parallel metaheuristics and identified a fast grown on these techniques in the last few years. The authors also classified the parallel frameworks and enumerated a few challenge topics that will be the main lines of research in the upcoming years.

According to Alba et al. [2013], parallel metaheuristics can be classified into *population-based* and *trajectory-based* methods. Population-based metaheuristics are methods that keep a pool of solutions (e.g. Genetic Algorithms and Ant Colony Optimization). Trajectory-based metaheuristics are methods that keep a single current solution (e.g. Simulated Annealing and Tabu Search). These two categories can also, be classified by the type of framework implementation [Alba et al., 2013].

Population-based metaheuristics can be classified into i) *Parallel individuals evaluation* – several individuals in the population are evaluated in parallel, and ii) *Parallel islands* – the population is split into subpopulations that can be managed in parallel.

Trajectory-based metaheuristics can be classified into i) *Parallel moves* – the neighboring solutions of the current solution are searched in parallel, ii) *Move acceleration* – the objective function of a single solution may be decomposed and evaluated in parallel, and iii) *Parallel multi-start* – several asynchronous threads, cooperative or independent, run simultaneously to search different regions of the solution space.

Among the hot topics for future research in parallel metaheuristics, Alba et al. [2013] point out the design of parallel versions of well-known metaheuristics that could benefit from specific hardware architectures. In this paper, we propose a parallel multi-start approach based on Iterated Local Search that is suitable for multi-core machines. The algorithm is designed to exploit diversification, intensification, and cooperation among asynchronous threads during the search. We also propose a constructive heuristic based on Relax-and-Fix [Dillenberger et al., 1994] and Proximity Search [Fischetti and Monaci, 2014] to provide initial solutions to our parallel approach.

The remainder of the paper is organized as following. Section 2 describes the addressed problem by a mixed-integer programming formulation. Section 3 explains our parallel approach. Section 4 presents and discusses our computational experiments. The paper ends in Section 5 with some final remarks.

The high school timetabling problem

We focus on an HSTP motivated by rules of Brazilian high schools. In this context, the schools have a set of classes and a set of teachers. Classes are disjoint groups of students enrolled in the same set of subjects (e.g.: mathematics, physics and etc.). Each subject has a pre-assigned teacher and a number of meetings that must be scheduled to run during the week (e.g. Monday to Friday). The goal of the problem is to obtain a weekly timetable for these meetings. An input of the problem is described by the following definitions:

Notation	Definition
Sets	
C	a set of classes.
T	a set of teachers.
D	a set of weekdays.
H	a set of periods per day, which is equal to every day.
H_{td}	the set of periods for which teacher $t \in T$ is available during day $d \in D$.
R	a set of subjects' requirements. A requirement specifies that a teacher $t \in T$ and a class $c \in C$ must meet a specified number of times during the week.
R_c	the set of requirements that belongs to class $c \in C$.
R_t	the set of requirements that belongs to teacher $t \in T$.
Parameters	
$\widetilde{\theta}_r \in \mathbb{N}$	the number of weekly meetings specified in requirement $r \in R$.
$\widetilde{\delta}_r \in \mathbb{N}$	a daily limit for lessons involving requirement $r \in R$.
$\widetilde{\pi}_r \in \mathbb{N}$	a desired number of double lessons for requirement $r \in R$. Double lessons are two lessons in consecutive periods of the same day.

An output of the problem is represented by a set of binary variables x_{rdh} which indicate whether the r-th requirement is scheduled to period $h \in H$ of day $d \in D$. A feasible solution is an assignment of values to variables x_{rdh} that respects the following hard requirements:

1. **Meeting of weekly required lessons**: each requirement must be scheduled.
$$\sum_{d \in D} \sum_{h \in H} x_{rdh} = \widetilde{\theta}_r \qquad \forall\, r \in R \tag{1}$$
2. **No clashes in classes' schedules**: each class must attend exactly one meeting per period.
$$\sum_{r \in R_c} x_{rdh} = 1 \qquad \forall\, c \in C;\ d \in D;\ h \in H \tag{2}$$
3. **No clashes in teachers' schedules**: each teacher must teach at most one lesson per period.
$$\sum_{r \in R_t} x_{rdh} \leq 1 \qquad \forall\, t \in T;\ d \in D;\ h \in H_{td} \tag{3}$$
4. **No assignment of teachers in their unavailable periods**: teachers must not be assigned to their unavailable periods.
$$\sum_{r \in R_t} x_{rdh} = 0 \qquad \forall\, t \in T;\ d \in D;\ h \in H \setminus H_{td} \tag{4}$$
5. **No daily workload violation for requirements**: each requirement $r \in R$ must not be involved in more than $\widetilde{\delta}_r$ meetings per day.
$$\sum_{h \in H} x_{rdh} \leq \widetilde{\delta}_r \qquad \forall\, r \in R;\ d \in D \tag{5}$$
6. **No holes in requirements' schedules**: the schedule of each requirement must be consecutive within the same day.
$$x_{rdi} - x_{rdh} + x_{rdj} - 1 \leq 0 \qquad \forall\, r \in R;\ d \in D;\ h = 1, \cdots, |H| - 2;\ i = 0, \cdots, h-1;\ j = h+1, \cdots, |H| - 1 \tag{6}$$

An optimal feasible solution minimizes the penalties associated with the following soft requirements:

7. **Meeting of double lessons for requirements**: for each requirement $r \in R$, the minimum number of $\widetilde{\pi}_r$ consecutive double lessons should be met. The auxiliary variables $\widehat{\phi}_{rdh}$ are set

to 1 if a double lesson ends in periods $h = 1, \cdots, |H| - 1$ within each day, while variables $\widehat{\pi}_r$ quantify the number of unmet weekly double lessons.

$$\widehat{\phi}_{rdh} \leq x_{rdh} \qquad \forall\, r \in R;\ d \in D;\ h = 1, \cdots, |H| - 1 \tag{7}$$

$$\widehat{\phi}_{rdh} \leq x_{r,d,h-1} \qquad \forall\, r \in R;\ d \in D;\ h = 1, \cdots, |H| - 1 \tag{8}$$

$$\widehat{\phi}_{rdh} \leq 1 - \widehat{\phi}_{r,d,h-1} \qquad \forall\, r \in R;\ d \in D;\ h = 2, \cdots, |H| - 1 \tag{9}$$

$$\widehat{\pi}_r \geq \widetilde{\pi}_r - \sum_{d \in D} \sum_{h=1}^{|H|-1} \widehat{\phi}_{rdh} \qquad \forall\, r \in R \tag{10}$$

$$\widehat{\phi}_{rdh} \geq 0 \qquad \forall\, r \in R;\ d \in D;\ h = 1, \cdots, |H| - 1 \tag{11}$$

$$\widehat{\pi}_r \geq 0 \qquad \forall\, r \in R \tag{12}$$

8. **No idle periods in teachers' schedules**: teachers should not have idle periods in their daily schedules. A teacher t is idle in periods $h = 1, \cdots, |H| - 2$ of a day d if he/she is not busy, but is busy in earlier and later periods of the same day. Idles periods are flagged by the auxiliary variables $\widehat{j}_{tdh}$.

$$\widehat{j}_{tdh} \geq \sum_{r \in R_t} (x_{rdi} - x_{rdh} + x_{rdj}) - 1 \qquad \forall\, t \in T;\ d \in D;\ h = 1, \cdots, |H| - 2;$$

$$i = 0, \cdots, h - 1;\ j = h + 1, \cdots, |H| - 1 \tag{13}$$

$$\widehat{j}_{tdh} \geq 0 \qquad \forall\, t \in T;\ d \in D;\ h = 1, \cdots, |H| - 2 \tag{14}$$

9. **Compact schedules for teachers**: for each teacher, the weekly schedule should encompass a minimum number of working days.

$$\widehat{d}_{td} \geq \sum_{r \in R_t} x_{rdh} \qquad \forall\, t \in T;\ d \in D;\ h \in H_{td} \tag{15}$$

$$\widehat{d}_{td} \geq 0 \qquad \forall\, t \in T;\ d \in D \tag{16}$$

The complete problem can thus, be described by the following mixed-integer programming (MIP) formulation.

$$\textbf{Minimize} \quad f(x) = \sum_{r \in R} \alpha_7 \widehat{\pi}_r + \sum_{t \in T} \sum_{d \in D} \alpha_9 \widehat{d}_{td} + \sum_{h=1}^{|H|-2} \alpha_8 \widehat{j}_{tdh} \tag{17}$$

Subject to:

$$(1) - (16) \tag{18}$$

$$x_{rdh} \in \{0, 1\} \qquad \forall\, r \in R;\ d \in D;\ h \in H \tag{19}$$

The objective function (17) minimizes the number of violations in requirements $i = 7, \cdots, 9$. The constant α_i is an associated penalty that express the importance of the i-th requirement.

This formulation can also be augmented by the cuts (20), proposed by Souza [2000], which specify that a teacher cannot work less than a minimum number of working days.

$$\sum_{d \in D} \widehat{d}_{td} \geq \max \left\{ \left\lceil \frac{\sum_{r \in R_t} \widetilde{\theta}_r}{|H|} \right\rceil,\ \max_{r \in R_t} \left\{ \left\lceil \frac{\widetilde{\theta}_r}{\widetilde{\delta}_r} \right\rceil \right\} \right\} \qquad \forall\, t \in T \tag{20}$$

The proposed approach

We propose an approach with constructive and improvement phases. The approach applies a heuristic based on Relax-and-Fix and Proximity Search (Section 3.1) to construct an initial feasible solution that is further improved by a Parallel Multi-Start Iterated Local Search (Section 3.2).

The constructive Proximity Relax-and-Fix heuristic (PRF)

The original Relax-and-Fix (RF) [Dillenberger et al., 1994] is an MIP heuristic that constructs integer solutions by solving a series of relaxed sub-problems. Let's assume that X is the set of integer variables of an MIP model. The RF heuristic relaxes the integrality of all variables of X and splits them in n subsets X^k, such that $X = \bigcup_{k=0}^{n-1} X^k$. At each iteration k, the variables belonging to subset X^k are converted to integer and the resulting problem is solved. If it results in a feasible solution, then the variables of X^k are fixed to their current values and the heuristic is repeated to iteration $k+1$. Otherwise, no solution is possible for the chosen partition and the heuristic stops. If the RF heuristic does not stop prematurely, it provides an integer solution at the end.

Algorithm 1 Pseudo-code of the constructive PRF heuristic.

PRF(m_0, m)

Let m_0 be the size of the first partition and m be the size of the remaining partitions.
Let P be the linear program of formulation (17) – (19).
Initialize P with constraints (1) – (6) and set $f(x) = \emptyset$.
Let LTS be the list of time-slots $(d, h) \in D \times H$.
Shuffle the list LTS.
for $i = 0$ **to** $m_0 - 1$ **do**
    Insert the variables associated with time-slot $LTS[i]$ into partition X^0.
$n = 1$
for $j = m_0$ **to** $|LTS| - 1$ **by** m **do** // compute the remaining partitions
    $n = n + 1$
    for $i = j$ **to** $\min\{j + m - 1,\ |LTS| - 1\}$ **do**
        Insert the variables associated with time-slot $LTS[i]$ into partition X^{n-1}.
Convert the variables of partition X^0 to integer.
Solve P.
Record the partial integer solution $S^0 = X^0$.
Fix the variables of partition X^0 to their current values.
Insert constraints (15) – (16) and the cuts (20) into P and set $f(x) = \alpha_9 \sum_{t \in T} \sum_{d \in D} \widehat{d}_{td}$.
for $k = 1$ **to** $n - 1$ **do**
    Convert the variables of partition X^k to integer.
    Solve P.
    if P is feasible **then**
        Record $S^k = S^{k-1} \cup X^k$.
        Fix the variables of partition X^k to their current values.
    else
        Estimate the cost of S^k, given by $f_k = (f(S^{k-1}) \div (m_0 + (k-1)m)) \cdot (m_0 + km)$.
        Release the variables of partition $\bigcup_{i=0}^{k-1} X^i$.
        Set P with the proximity function $\Delta(x, \widetilde{x}) = \sum_{j \in S^{k-1}: \widetilde{x}_j = 0} x_j + \sum_{j \in S^{k-1}: \widetilde{x}_j = 1} (1 - x_j)$.
        Insert the cut-off constraint $f(x) \leq f_k$ into P.
        Solve P.
        if P is infeasible **then**
            return $\emptyset$.
        Record $S^k = \bigcup_{i=0}^{k} X^i$.
        Fix the variables of partition $\bigcup_{i=0}^{k} X^i$.
        Remove the cut-off constraint and set P with the objective function $f(x) = \alpha_9 \sum_{t \in T} \sum_{d \in D} \widehat{d}_{td}$.
return S^{n-1}.

The drawback of the original RF heuristic, when applied to the HSTP described in Section 2, is that the fixation of a partition X^k, at iteration k, very often leads to an infeasible problem at iteration $k+1$. To overcome this limitation, we propose a variant of the RF heuristic which solves a proximity problem [Fischetti and Monaci, 2014] to escape from these infeasible partitions. Silva [2013] proposed a similar strategy to deal with this limitation of the RF heuristic in lot-sizing problems. The author employs a local branch constraint instead of a proximity model to find new

feasible partitions and escape from infeasibilities.

The proposed heuristic (PRF) is shown in Algorithm 1. The algorithm is a relax-and-fix strategy with the addition of lines 25 to 34, that solve the proximity problem to escape from infeasible partitions. At each iteration k, every time that the algorithm detects an infeasible partition, the line 25 estimates the cost f_k of the partial solution S^k, based on the previous solution S^{k-1}. The line 26 releases all fixed variables and lines 27 to 29 set and solve the proximity problem. The proximity problem tries to find a partial solution S^k that is not greater than f_k and minimizes the changes in the values previously fixed for variables of S^{k-1}. If the proximity problem is feasible, a new partial solution S^k is found, all variables in the partition $\bigcup_{i=0}^{k} X^i$ are fixed to their current values and the algorithm continues in the next iteration. Otherwise, it stops without any solution.

In our implementation, each partition contains the variables associated with a subset of time-slots $(d, h) \in D \times H$. We defined the first partition with size $m_0 = 10$ time-slots and the rest with $m = 5$ time-slots. Also, every time that the program P is solved in lines 14, 20 and 29, we stop it at the first solution found, to speed up the execution.

The Parallel Multi-Start Iterated Local Search (PMILS)

The proposed Parallel Multi-Start Iterated Local Search (PMILS), shown in Figure 1, employs a shared-memory manager/works strategy. The central idea of the PMILS is to exploit diversification/intensification and cooperation among threads. The algorithm has a manager thread (the diversifier agent) that shares information with N worker threads (the intensifier agents) via two communication buffers, "$InputBuffer$" and "$OutputBuffer$". All threads are Iterated Local Search [Lourenço et al., 2003] metaheuristics.

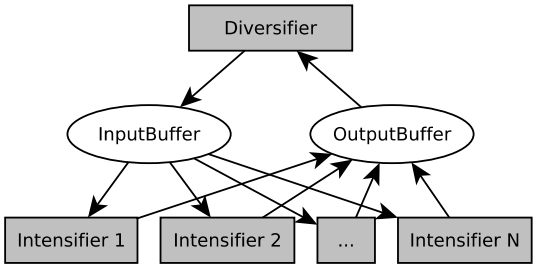

Figure 1: Diagram of communication among the agent threads of the PMILS algorithm.

The PMILS pseudo-code is shown in Algorithm 2. The diversifier agent starts at an initial solution and at each iteration, puts its local minimum into the input buffer (lines 10–14) to be further explored by the intensifier agents. The intensifiers get solutions from the input buffer (line 3) and intensify the search in their nearby neighborhoods (line 6). When an intensifier reaches its time-limit $intTimeLim$, it puts its best solution into the output buffer (line 7) to be analyzed by the diversifier. When the diversifier retrieves a solution S_r from the output buffer (line 15), it checks for acceptance. If S_r improves its best solution S^*, then the diversifier restarts its search from S_r (line 19). Otherwise, S_r is discarded.

The difference between the diversifier and the intensifier threads is that, while the diversifier walks through the solution space (lines 6–7) by only restarting from the global best solution when an improved solution is found (line 19). The intensifiers explore the nearby neighborhood of the local minima found by the diversifier, by always restarting from their private best solutions, see line 3 of the ILS procedure.

The timetable solutions used by the PMILS algorithm are represented by non-negative integer arrays S, where S_{cdh} stores the teacher $t \in T$ assigned to teach to class $c \in C$ on period $h \in H$ of day $d \in D$. The neighborhood operator is the TQ operator proposed by Saviniec et al. [2013], which is based on the idea of *Kempe Chain Interchanges* [Lü et al., 2011] and explores moves by exchanging conflicting teachers between two time-slots $(d,\ h) \in TS$, with $TS = D \times$

H. The local search procedure is a first improvement strategy that explores all pairs of time-slots $(p,\ q) \in TS \times TS$, for $p \neq q$.

Algorithm 2 Pseudo-code of the PMILS algorithm.

DIVERSIFIER$(S, divTimeLim, N)$

```
1  Initialize two buffers InputBuffer and OutputBuffer of size N.
2  Initialize N intensifier threads.
3  S* = S,  S_r = ∅
4  time = 0,  added = 0
5  while (time < divTimeLim) do
6      S = PERTURBATION(S, 2)                    // Apply two random moves
7      S = LOCALSEARCH(S)
8      if f(S) ≤ f(S*) then
9          S* = S
10     gap = 1 − max{f(S) − f(S*), 0} ÷ f(S*)
11     if GETRANDOM(0, 1) ≤ gap then
12         added = added + InputBuffer.TryAdd(S)
13     else
14         added = added + InputBuffer.TryAdd(S*)
15     removed = OutputBuffer.TryRemove(S_r)
16     added = added − removed
17     if removed then
18         if f(S_r) < f(S*) then
19             S = S_r                           // Restart from the global best solution
20         if f(S_r) ≤ f(S*) then
21             S* = S_r
22     time = CPUTIME()
23 while (added > 0) do
24     OutputBuffer.Remove(S_r)
25     added = added − 1
26     if f(S_r) < f(S*) then
27         S* = S_r
28 for k = 1 to N do
29     InputBuffer.Add(NULL)
30 BARRIERWAIT()
31 return S*
```

INTENSIFIER$(intTimeLim)$

```
1  S = ∅
2  while (true) do
3      InputBuffer.Remove(S)
4      if S = NULL then
5          break
6      S = ILS(S, intTimeLim)
7      OutputBuffer.Add(S)
8  return
```

ILS$(S_0, ilsTimeLim)$

```
1  S* = S_0,  time = 0
2  while (time < ilsTimeLim) do
3      S = PERTURBATION(S*, 1)                   // Apply one random move
4      S = LOCALSEARCH(S)
5      if f(S) ≤ f(S*) then
6          S* = S
7      time = CPUTIME()
8  return S*
```

Computational experiments

We tested the proposed approach with two variants of the problem described in Section 2. The first variant HSTP-A is the problem introduced by Souza [2000], for which we show the results in Section 4.1. The second variant HSTP-B is a modification of the problem proposed by Souza [2000], for which we show the results in Section 4.2. The algorithms were coded in C++ and compiled with GCC on Linux operating system. To implement parallelism, we employed the Pthreads library available in the GCC compiler. The MIP model and the PRF heuristic were implemented with callable libraries of CPLEX 12.6.

Results in the HSTP-A

This problem was introduced by Souza [2000]. The problem is a relaxation of the problem described in Section 2, for which hard constraints (6) are dropped.

In this experiment, we tested the proposed approach in the set of seven instances[1] proposed by Souza [2000]. The instances, shown in Table 1, have optimal solutions known, which is a contribution of several previous researches using heuristic solutions [Souza et al., 2003; Santos et al., 2005; Saviniec et al., 2013; Dorneles et al., 2014] and column generation lower bounds [Santos et al., 2012]. According to Santos et al. [2005], the objective function weighting parameters for these instances are: $\alpha_7 = 1$, $\alpha_8 = 3$, $\alpha_9 = 9$. For this experiment, we set the algorithm PMILS with parameters $N = 3$ intensifiers, $intTimeLim = 2$ seconds and total execution time (the two phases) of 625 seconds.

Table 1: Features of the instances proposed by Souza [2000].

ID	$\|C\|$	$\|T\|$	$\|D\|$	$\|H\|$	$\sum_{r\in R}\widetilde{\theta}_r$	$\sum_{r\in R}\widetilde{\pi}_r$	Optimal values
1	3	8	5	5	75	21	202
2	6	14	5	5	150	29	333
3	8	16	5	5	200	4	423
4	12	23	5	5	300	41	652
5	13	31	5	5	325	71	762
6	14	30	5	5	350	63	756
7	20	33	5	5	500	84	1017

Table 2: Results of 25 runs of our approach compared to previous methods in the instances of Souza [2000].

	PRF		PMILS			ILS		F8	
ID	Time (s)	Median	Time (s)	Median	Best	Time (s)	Best	Time (s)	Best
1	0.12	358	7.00	**202**	**202**	900	**202**	600	**202**
2	0.56	668	11.93	**333**	**333**	900	**333**	600	**333**
3	0.77	700	529.01	426	**423**	900	**423**	21600	**423**
4	6.04	824	19.00	**652**	**652**	900	**652**	600	**652**
5	4.27	1345	9.87	**762**	**762**	900	**762**	600	**762**
6	3.08	1425	13.80	**756**	**756**	900	**756**	1800	759
7	13.29	1872	18.23	**1017**	**1017**	900	**1017**	1800	**1017**

The results are shown in Table 2. The table compares the results of our approach with best results of two previous approaches. The Iterated Local Search (ILS) proposed by Saviniec et al. [2013] and the best version (F8) of the Fix-and-Optimize heuristics proposed by Dorneles et al. [2014]. The best solutions are shown in bold font. We observe that our PRF heuristic is very fast to construct initial solutions. The constructed solutions have an average optimality gap of 41.45 %. The PMILS algorithm found similar results in quality of solutions compared with the two other approaches. However, PMILS outperforms the previous approaches in computational time. It finds the same quality of solutions much faster than the previous approaches[2].

Results in the HSTP-B

This is the problem described in Section 2. In this problem, we used the 34 instances proposed in Saviniec et al. [2015]. The instances are described in Table 3. The objective function weighting parameters were set to $\alpha_7 = 1$, $\alpha_8 = 3$, $\alpha_9 = 9$.

In this experiment, our approach is compared with the solver GOAL [Fonseca et al., 2014], that is the winner of the Third International Timetabling Competition [Post et al., 2016] devoted to high school timetabling problems, ITC2011[3]. We compare our approach with the latest version of GOAL [Fonseca et al., 2016], which employs a three-phase approach. The first phase constructs an initial solution via the KHE software libraries [Kingston, 2015]. The second phase

[1] Available at the website "http://labic.ic.uff.br/Instance/index.php?dir= SchoolTimetabling"

[2] Our approach was run in a machine with inferior hardware resources than the machines described by Saviniec et al. [2013] and Dorneles et al. [2014]. We used a Notebook with CPU Intel Core i3 (2.3 GHz) and 2 GB of RAM, running *Linux Mint 17.1*.

[3] https://www.utwente.nl/ctit/hstt/

employs a parallel Variable Neighborhood Search (PVNS) metaheuristic and the third phase refines the output of PVNS with Fix-and-Optimize MIP heuristics. The experiment was made on a server with 2 CPU Intel Xeon E5-2680v2 (2.8 GHz) and 128 GB of RAM, running *Red Hat Enterprise Linux 6.5*. We set the algorithm PMILS with $N = 19$ intensifiers and the parallel VNS of GOAL with 20 threads. The additional parameters of GOAL were set with recommended values (alg-timelimit = 62, form-timelimit = 62, initial-soln = KHE, algorithm = SVNS, formulation = FIXOPT, formfixopt-nresources = 5 and formfixopt-optinarow = 5).

The results are shown in Table 4. The best solutions are shown in bold font. We observe that our approach outperforms GOAL in quality of solutions for most of the tested instances. The algorithm PMILS only performed worse than GOAL for instance 8. This instance has a large number of teachers' unavailable periods, which considerably reduces the feasible region and it is expected that the Fix-and-Optimize MIP heuristics implemented in GOAL perform better than our local search. The columns LB and UB of Table 4 show the results of CPLEX after a time-limit of 3 hours. The CPLEX was initialized with a solution found by running the PMILS algorithm during 625 seconds. The CPLEX was able to improve the solution of PMILS only for the restricted instance 8. The last column of Table 4 presents the best gaps found for these instances.

In this experiment, the PRF heuristic was also able to construct initial feasible solutions very fast. The constructed solutions have an average optimality gap of 37.89 %.

Table 3: Features of the instances proposed by Saviniec et al. [2015].

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 4: Comparison between our approach and the solver GOAL [Fonseca et al., 2016] in 25 runs of 625 seconds.

	PRF		PMILS		GOAL		CPLEX (3 h)		
ID	Time (s)	Median	Median	Best	Median	Best	LB	UB	Gap (%)
1	1.74	1158	710	**703**	730	716	682	–	2.99
2	1.94	1146	713	**706**	730	716	682	–	3.40
3	1.54	1514	751	**746**	770	762	726	–	2.68
4	1.19	1583	743	**738**	768	749	725	–	1.76
5	0.65	1253	**631**	**631**	648	642	629	–	0.32
6	1.84	1521	777	**773**	814	800	771	–	0.26
7	6.34	1583	1256	**1245**	1312	1297	1203	–	3.37
8	1.53	805	697	678	682	**676**	675	675	0.00
9	2.53	1002	822	**816**	838	829	804	–	1.47
10	0.29	688	**298**	**298**	300	**298**	298	–	0.00
11	3.64	1391	999	**978**	1052	1024	953	–	2.56
12	5.39	1184	1037	**1028**	1048	1034	1014	–	1.36
13	0.36	281	**273**	**273**	**273**	**273**	273	–	0.00
14	4.64	1098	950	**943**	971	949	929	–	1.48
15				Infeasible					
16	0.61	870	483	**481**	498	490	475	–	1.25
17	0.4	911	461	**458**	468	462	455	–	0.66
18	0.31	762	**319**	**319**	324	**319**	316	–	0.94
19	0.21	591	**254**	**254**	**254**	**254**	251	–	1.18
20	0.3	784	**325**	**325**	328	**325**	312	–	4.00
21	7.65	1266	1067	**1061**	1107	1091	1044	–	1.60
22				Infeasible					
23	11.8	2519	1893	**1865**	2013	1971	1793	–	3.86
24	11.5	2665	2065	**2047**	2171	2128	1965	–	4.01
25	4.02	1333	940	**924**	987	966	909	–	1.62
26	1.34	860	575	**572**	603	584	570	–	0.35
27	1.27	675	554	**552**	566	560	551	–	0.18
28	2.63	2009	1153	**1138**	1183	1167	1103	–	3.08
29	2.66	1958	1145	**1133**	1187	1168	1094	–	3.44
30	2.82	2041	1166	**1153**	1203	1175	1112	–	3.56
31	3.11	1853	1164	**1147**	1195	1181	1111	–	3.14
32	1.97	1928	1043	**1033**	1069	1063	998	–	3.39
33	2.12	1901	1050	**1038**	1078	1060	997	–	3.95
34	2.32	1811	1041	**1029**	1070	1056	979	–	4.86

Conclusions

In this paper, we proposed an approach with constructive and improvement phases to solve high school timetabling problems. The approach applies a Proximity Relax-and-Fix heuristic to construct an initial feasible solution that is further improved with a Parallel Multi-Start Iterated Local Search. Our constructive heuristic solves a series of relaxed sub-problems to achieve a completely feasible solution. When infeasible partitions are chosen, the heuristic is able to step over by solving a proximity problem to find another feasible partition. Our improvement heuristic is designed to exploit the parallelism of multi-core machines. The algorithm is a system of cooperative threads that exploit diversification and intensification to achieve high-quality solutions during the search. The proposed approach showed to find good quality solutions and outperform the state-of-the-art algorithms for two variants of the problem at hand. These results show that parallel cooperative multi-start approaches, such the one proposed here, are promising tools for handling combinatorial problems such as the HSTP. Also, these approaches are scalable algorithms that can exploit the power of machines with a large number of cores.

Acknowledgments

This research was supported by FAPESP-Brazil. Grants: 2013/13563-3 and 2015/10032-2. We would like to thank professor George H. G. Fonseca for sharing the code of GOAL.

References

Abramson, D. (1991). Constructing school timetables using simulated annealing: sequential and parallel algorithms. *Management science*, 37(1):98–113.

Abramson, D. and Abela, J. (1992). A parallel genetic algorithm for solving the school timetabling problem. *Proceedings of the 15 Australian Computer Science Conference, Hobart, Australia*, p. 1–11.

Alba, E., Luque, G., and Nesmachnow, S. (2013). Parallel metaheuristics: recent advances and new trends. *International Transactions in Operational Research*, 20(1):1–48.

Di Gaspero, L., McCollum, B., and Schaerf, A. (2007). The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). Technical report, Technical Report QUB/IEEE/Tech/ITC2007/CurriculumCTT/v1. 0, Queen's University, Belfast, United Kingdom.

Dillenberger, C., Escudero, L. F., Wollensak, A., and Zhang, W. (1994). On practical resource allocation for production planning and scheduling with period overlapping setups. *European Journal of Operational Research*, 75(2):275 – 286. ISSN 0377-2217.

Dorneles, Á. P., de Araújo, O. C., and Buriol, L. S. (2014). A fix-and-optimize heuristic for the high school timetabling problem. *Computers & Operations Research*, 52, Part A(0):29 – 38.

Fischetti, M. and Monaci, M. (2014). Proximity search for 0-1 mixed-integer convex programming. *Journal of Heuristics*, 20(6):709–731. ISSN 1572-9397.

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

Fonseca, G. H., Santos, H. G., Toffolo, T. A., Brito, S. S., and Souza, M. J. (2014). Goal solver: a hybrid local search based solver for high school timetabling. *Annals of Operations Research*, p. 1–21. ISSN 0254-5330.

Kingston, J. H. (2015). A software library for high school timetabling. URL `http://sydney.edu.au/engineering/it/~jeff/khe/`.

Lewis, R., Paechter, B., and McCollum, B. (2007). *Post enrolment based course timetabling: A description of the problem model used for track two of the second international timetabling competition*. Cardiff Business School.

Lourenço, H., Martin, O., and Stützle, T. (2003). Iterated local search. *Handbook of metaheuristics*, p. 320–353.

Lü, Z., Hao, J., and Glover, F. (2011). Neighborhood analysis: a case study on curriculum-based course timetabling. *Journal of Heuristics*, 17(2):97–118.

McCollum, B., McMullan, P., Burke, E. K., Parkes, A. J., and Qu, R. (2007). The second international timetabling competition: Examination timetabling track. Technical report, Technical Report QUB/IEEE/Tech/ITC2007/-Exam/v4.0/17, Queen's University, Belfast, United Kingdom.

Pillay, N. (2014). A survey of school timetabling research. *Annals of Operations Research*, 218(1): 261–293. ISSN 1572-9338.

Post, G., Di Gaspero, L., Kingston, J. H., McCollum, B., and Schaerf, A. (2016). The third international timetabling competition. *Annals of Operations Research*, 239(1):69–75.

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

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

Saviniec, L., Constantino, A. A., Romão, W., and Santos, H. G. (2013). Solving the high school timetabling problem to optimality by using ils algorithms. *Proceedings of the Brazilian Symposium on Operations Research, Sobrapo, Rio de Janeiro, Brazil*, p. 3330–3341.

Saviniec, L., Santos, M. O., Costa, A. M., and Constantino, A. A. (2015). Multithreading iterated local search aplicado ao problema de horários escolares. *Proceedings of the Brazilian Symposium on Operations Research, Sobrapo, Rio de Janeiro, Brazil*, p. 826–837.

Silva, D. H. (2013). Métodos híbridos para o problema de dimensionamento de lotes com múltiplas plantas. Master's thesis, Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional - Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo, São Carlos.

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

Souza, M., Ochi, L., and Maculan, N. (2003). A grasp-tabu search algorithm for solving school timetabling problems. *Metaheuristics: Computer Decision-Making. Kluwer Academic Publishers, Boston*, p. 659–672.

Srndic, N., Pandzo, E., Dervisevic, M., and Konjicija, S. (2009). The application of a parallel genetic algorithm to timetabling of elementary school classes: A coarse grained approach. In *Information, Communication and Automation Technologies, 2009. ICAT 2009. XXII International Symposium on*, p. 1–5. IEEE.