Late acceptance hill-climbing for high school timetabling

3 authors:

George H.G. Fonseca
Universidade Federal de Ouro Preto

11 PUBLICATIONS **48** CITATIONS

SEE PROFILE

Haroldo Gambini Santos
Universidade Federal de Ouro Preto

46 PUBLICATIONS **302** CITATIONS

SEE PROFILE

Eduardo G. Carrano
Federal University of Minas Gerais

70 PUBLICATIONS **495** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project Optimization applied to communication network design and operation View project

Project Combinatorial optimization View project

Late Acceptance Hill-Climbing for High School Timetabling

George H.G. Fonseca · Haroldo G. Santos · Eduardo G. Carrano

Abstract The application of the Late Acceptance Hill-Climbing (LAHC) to solve the High School Timetabling Problem is the subject of this manuscript. The original algorithm and two variants proposed here are tested jointly with other state-of-art methods to solve the instances proposed in the Third International Timetabling Competition. Following the same rules of the competition, it has been noticed that the LAHC based algorithms outperformed the winner methods. These results, and reports from the literature, suggest that the LAHC is a reliable method that can compete with the most employed local search algorithms.

Keywords Late Acceptance Hill-Climbing · Third International Timetabling Competition · High School Timetabling · Local Search

1 Introduction

The High School Timetabling Problem (HSTP) is faced by many educational institutions around the world. A solution for this problem consists in an assignment of timeslots and resources to the events, respecting several constraints. Generally, this assignment is repeated weekly, until the end of the semester. Beyond its practical importance, this problem is $\mathcal{NP}$-Hard [7], which justifies the intense efforts dedicated by the Operations Research and Computational Intelligence communities in proposing methods for solving it [4, 19, 23].

The problem relevance and complexity motivated the organization of three International Timetabling Competitions (ITC), in which the researchers could test

George H. G. Fonseca and Eduardo G. Carrano
Graduate Program in Electrical Engineering
Federal University of Minas Gerais
Av. Antônio Carlos 6627, 31270-901 Belo Horizonte, MG, Brazil
E-mail: `george@decsi.ufop.br`, `egcarrano@ufmg.br`

Haroldo G. Santos
Department of Computing
Federal University of Ouro Preto
St. Diogo de Vasconcelos 328, 35400-000 Ouro Preto, MG, Brazil
E-mail: `haroldo@iceb.ufop.br`

their approaches in the same computational environment. The first competition (ITC2003) [9] was won by Kostuch [16], with a 3-phase local search-based algorithm. The second one (ITC2007) [18] was won by Muller [20], with a variation of the Simulated Annealing algorithm (SA). The last one (ITC2011) [18] was won by Fonseca *et al.* [6], who proposed a hybrid approach combining Simulated Annealing and Iterated Local Search (namely SA-ILS).

The results of the competitions indicated that the local search methods are currently leading to the best results of the HSTP. Among these methods, it is possible to highlight the Simulated Annealing algorithm [15], which was present in the three competition winners. Approaches based on Integer Programming were also proposed [17,26], but they are restricted to small instances due to their computational complexity.

A new technique, applied to the HSTP model introduced in ITC2011, is proposed in this paper. This technique is based on a local search method called Late Acceptance Hill-Climbing (LAHC), proposed by Burke and Bykov [3]. This algorithm was modified, generating two new variants that intend to overcome some limitations of the original method. The new algorithms obtained remarkable results, outperforming the best marks from the ITC2011 winner. Moreover, the implementation of the algorithm is quite easy and, up to the authors belief, it can be extended to other combinatorial optimization problems without major adaptations.

The outline of this paper is structured as follows. Section 2 presents the model of the HSTP adopted in ITC2011. The proposed solution approaches are presented in Section 3. Results for computational experiments are given in Section 4. Finally, concluding remarks are drawn in Section 5.

2 High School Timetabling Problem Model

The Third International Timetabling Competition motivated the development of methods for solving hard school timetabling problems. It also encouraged the alignment of research and practice, making real-world instances available. The organizers also provided a benchmark to adjust processing times and a solution validator.

The instances were specified in the XHSTT format, which is a XML (eXtensible Markup Language) based format adapted to describe timetabling problems. Post *et al.* [24] also highlighted that this format can specify instances of other timetabling problems, beyond the scholar context.

The considered model of High School Timetabling Problem came up with the goal of providing a generic model capable of addressing various features of the HSTP in real world situations [8,12,21,24,26,29,31]. The model is split in four main entities, which are described in the following subsections.

2.1 Times

The times entity consists of a single TIME (timeslot) or a set of times, called TIME GROUP. The timeslots are commonly grouped by DAY (e.g. timeslots of Monday).

2.2 Resources

The resources entity consists of a single RESOURCE, a set of resources (RESOURCE GROUP) or a RESOURCE TYPE. Each single resource belongs to a specific resource type. In the context of school timetabling, the most common resource types are [24]:

CLASS: a group of students who attend the same events. Important constraints to the classes are to control idle times and the number of lessons per day;

TEACHER: a teacher can be preassigned to attend an event. In some cases, pre-assignment is not possible, and the teacher should be assigned according to its qualifications and workload limits;

ROOM: most events take place in a room. One room has a certain capacity and a set of features.

2.3 Events

An EVENT usually represents a set of lessons about a subject. It demands a set of times and resources to occur. This assignment is the main goal of any timetabling solver. Events may be grouped into an EVENT GROUP. A timeslot assigned to an event is called MEET and a resource assigned to an event is called TASK. Every XHSTT solver is also responsible to break an event in sub-events to be spread over the days whenever it is necessary. Other kinds of events, like meetings, are allowed by the model [24]. An event has the following attributes:

- *duration*, which represents the number of times that should be assigned to an event;
- *workload*, which will be added to the total workload of resources assigned to the event (optional);
- *preassigned resources* to attend the event (optional);
- *preassigned timeslots* to attend the event (optional).

2.4 Constraints

Post *et al.* [24] groups the CONSTRAINTS into three categories: basic scheduling constraints, event constraints and resource constraints. The OBJECTIVE FUNCTION $f(.)$ is calculated in terms of violations of the constraints. These violations are penalized according to the WEIGHT of each constraint, defining a minimization problem. They are also divided into hard constraints, whose attendance is mandatory; and soft constraints, whose attendance is desirable. Each instance can define whether a constraint is hard or soft and its weight. For more details, please refer to Post *et al.* [24]. A mathematical programming formulation of all XHSTT constraints is given by Kristiansen *et al.* [17].

2.4.1 Basic Scheduling Constraints

- ASSIGN TIME: assigns the required number of timeslots to each event;
- ASSIGN RESOURCE: assigns the required resources to each event;

– PREFER TIMES: indicates that some events have preference for particular times-
  lots;
– PREFER RESOURCES: indicates that some events have preference for particular
  resources.

2.4.2 Event Constraints

– LINK EVENTS. Schedules a set of events to the same timeslots;
– SPREAD EVENTS. Specifies that the number of occurrences of an event in a
  timeslot group should lie between a minimum and a maximum value. This
  constraint can be used, for example, to define a daily limit of lessons of a given
  subject;
– AVOID SPLIT ASSIGNMENTS. Assigns the same resources to all occurrences of the
  same event. With this constraint, for example, one can enforce the assignment
  of all occurrences of an event to the same room;
– DISTRIBUTE SPLIT EVENTS. For each event, assigns between a minimum and a
  maximum sub-events of a given duration. This constraint may be important
  in some institutions, since a large number of consecutive lessons of the same
  subject can affect the performance of the students;
– SPLIT EVENTS. Limits the number of non-consecutive meets that an event
  should be scheduled and its duration. One example of an application of this
  constraint is to ensure that an event of duration four is split into two sub-events
  of duration two.

2.4.3 Resource Constraints

– AVOID CLASHES. Assigns the resources without clashes (i.e. without assigning
  the same resource in more than one event at a given time);
– AVOID UNAVAILABLE TIMES. States that some resources are unavailable to at-
  tend any event at certain times. For instance, this constraint can be used to
  avoid that a teacher is assigned to timeslot that he/she could not attend;
– LIMIT WORKLOAD. Restricts the workload of the resources between minimum
  and maximum bounds;
– LIMIT IDLE TIMES. Sets that the number of idle times in each time group should
  lie between a minimum and a maximum bound for each resource. Typically, a
  time group consists of all timeslots of a given day of the week. This constraint
  is used to avoid inactive timeslots between active ones in the schedule of a
  given resource;
– LIMIT BUSY TIMES. The number of busy times in each day should lie between
  minimum and maximum bounds for each resource. A high number of alloca-
  tions in the same day can affect student and teacher performances;
– CLUSTER BUSY TIMES. The number of time groups with a timeslot assigned
  to a resource should lie between minimum and maximum limits. This can be
  used, for example, to concentrate teacher's activities into a minimum number
  of days.

3 Solution Approach

The proposed approach is composed of two main steps: (*i*) an initial solution is generated using the Kingston High School Timetabling Engine (KHE) constructive algorithm [13]; (*ii*) this solution is used as a starting point for the Late Acceptance Hill-Climbing (LAHC) metaheuristic, or one of our proposed variants, in order to find improved solutions using multi-neighbourhood local search. These elements are explained in the next subsections.

3.1 Build Method

The Kingston High School Timetabling Engine is a platform for handling instances of the addressed problem. It also provides a solver, which has been used to generate initial solutions because it can find solutions of reasonable quality in a short time [14]. A very brief description of KHE will be given in the next paragraphs. For more details, please refer to [11,14].

KHE generates a solution through a three step approach. The first one is the *structural phase*. It constructs an initial solution with no time or resource assignments and it creates structures for the next phases. The structural phase splits events into sub-events whose durations depend on constraints related to how events should be split (namely, split events, distribute split events and spread events), and groups the sub-events (so called *meets*) into sets called *nodes*. Sub-events derived from the same event goes into the same node. Sub-events whose original events are connected by a spread events or avoid split assignments constraint also lie in the same node. Events connected by link events constraints have their meets connected in such a way that whenever a time is assigned to one of these meets, this assignment is also extended to the other connected meets. Each meet also contains a set of times called *domain*. Only times from this set may be assigned to the meet. Domains are chosen based on prefer times constraints. A meet contains one *task* for each demanded resource in the event that it was derived from. Each task also contains a set of resources of the proper type called *domain*. When the resource is pre-assigned, the domain contains only the pre-assigned resource, otherwise, this domain is based on prefer resource constraints. This step also assigns pre-assigned times and resources.

Next comes the *time assignment phase*, which assigns a time to each meet. For each resource to which a hard avoid clashes constraint applies it builds a layer - the set of nodes containing meets preassigned to that resource. After merging layers wherever one nodes are a subset of the other, and sorting them in such a way that the most difficult layers (with fewer available choices for assignment) come first, it assigns times to the meets of each layer. This assignment is made through a minimum-cost matching between meets of the given layer and times. Each edge of such a graph has a cost according to the objective function cost of this assignment.

Finally comes the *resource assignment phase*. For each resource type, an iteration of the following procedure is performed. If the resource assignment for this resource type is constrained by avoid split assignments constraint, a resource packing algorithm is invoked. Otherwise, a simple heuristic is used. A packing of a resource consists in finding assignments of tasks to the resource that makes the

solution cost as small as possible, using the resource as much as possible under its workload limits. The resources are placed in a priority queue in which the most demanded are prioritized. At each iteration, a resource is dequeued and processed. The packing procedure consists of a simple binary tree search over the elective tasks of a given resource. The simple heuristic consists in, for each task from the most constrained to the least one, assigning the resource that provides more improvement on the objective function. It is possible to estimate the amount of tasks whose resource assignment is impossible (ideally 0). This is performed through a maximum matching in an unweighed bipartite graph, where tasks are demand nodes and resources are supply nodes. This estimate is called *resource assignment invariant* and it is kept minimal through the whole resource assignment process.

3.2 Neighborhood Structure

The neighborhood structure $N(s)$ considered in the proposed methods is composed of six types of moves[1]. This neighborhood structure is very similar to the one proposed by the winner of ITC2011 [5,6], except that the move Permute Resources was removed. This move is computationally expensive and it was not contributing significantly to achieve good solutions. The considered moves are presented in the following subsections.

3.2.1 Event Swap (ES)

Two events e_1 and e_2 are selected and have their timeslots t_1 and t_2 swapped. Figure 1 presents an example of this move.

Mon	Tue	Wed	Thu	Fri
Math$_1$	Eng$_1$	Math$_4$	Phis$_1$	Eng$_3$
Math$_2$	Eng$_2$	Math$_5$	Phis$_2$	Eng$_4$
Math$_3$	Chem$_1$	Geog$_3$	Span$_1$	Eng$_5$
Geog$_1$	Chem$_2$	His$_1$	Span$_2$	Phis$_3$
Geog$_2$	Chem$_3$	His$_2$	His$_3$	

ES($Geog_3$, Eng_5)

Mon	Tue	Wed	Thu	Fri
Math$_1$	Eng$_1$	Math$_4$	Phis$_1$	Eng$_3$
Math$_2$	Eng$_2$	Math$_5$	Phis$_2$	Eng$_4$
Math$_3$	Chem$_1$	Eng$_5$	Span$_1$	Geog$_3$
Geog$_1$	Chem$_2$	His$_1$	Span$_2$	Phis$_3$
Geog$_2$	Chem$_3$	His$_2$	His$_3$	

Fig. 1 Example of Event Swap [6].

3.2.2 Event Move (EM)

An event e_1 is moved from its original timeslot t_1 to a new timeslot t_2. Figure 2 presents an example of this move.

---

[1] We denote by $N_k(s)$ the subset of $N_k(s)$ involving only moves of type k.

Mon	Tue	Wed	Thu	Fri
Math$_1$	Eng$_1$	Math$_4$	Phis$_1$	Eng$_3$
Math$_2$	Eng$_2$	Math$_5$	Phis$_2$	Eng$_4$
Math$_3$	Chem$_1$	Geog$_3$	Span$_1$	Eng$_5$
Geog$_1$	Chem$_2$	His$_1$	Span$_2$	Phis$_3$
Geog$_2$	Chem$_3$	His$_2$	His$_3$	

EM(*Chem$_3$, Fri_5*)

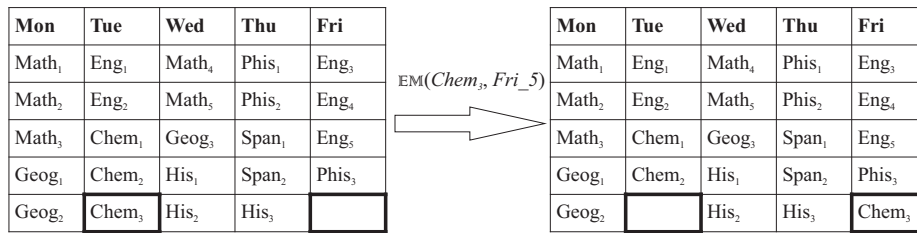| Mon	Tue	Wed	Thu	Fri
Math$_1$	Eng$_1$	Math$_4$	Phis$_1$	Eng$_3$
Math$_2$	Eng$_2$	Math$_5$	Phis$_2$	Eng$_4$
Math$_3$	Chem$_1$	Geog$_3$	Span$_1$	Eng$_5$
Geog$_1$	Chem$_2$	His$_1$	Span$_2$	Phis$_3$
Geog$_2$		His$_2$	His$_3$	Chem$_3$

Fig. 2 Example of Event Move [6].

*3.2.3 Event Block Swap (*EBS*)*

Similarly to ES move, the Event Block Swap swaps the timeslots of two events e_1 and e_2, but, when the events have different durations, e_1 is moved to the last time-slot occupied by e_2. This move allows timeslot swaps without losing the allocation contiguity. Figure 3 presents an example of this move.
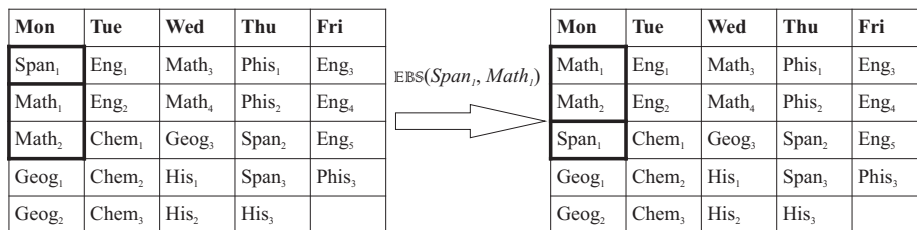
Mon	Tue	Wed	Thu	Fri
Span$_1$	Eng$_1$	Math$_3$	Phis$_1$	Eng$_3$
Math$_1$	Eng$_2$	Math$_4$	Phis$_2$	Eng$_4$
Math$_2$	Chem$_1$	Geog$_3$	Span$_2$	Eng$_5$
Geog$_1$	Chem$_2$	His$_1$	Span$_3$	Phis$_3$
Geog$_2$	Chem$_3$	His$_2$	His$_3$	

EBS(*Span$_1$, Math$_1$*)

Mon	Tue	Wed	Thu	Fri
Math$_1$	Eng$_1$	Math$_3$	Phis$_1$	Eng$_3$
Math$_2$	Eng$_2$	Math$_4$	Phis$_2$	Eng$_4$
Span$_1$	Chem$_1$	Geog$_3$	Span$_2$	Eng$_5$
Geog$_1$	Chem$_2$	His$_1$	Span$_3$	Phis$_3$
Geog$_2$	Chem$_3$	His$_2$	His$_3$	

Fig. 3 Example of Event Block Swap [6].

*3.2.4 Resource Swap (*RS*)*

Two events e_1 and e_2 have their assigned resources r_1 and r_2 swapped. Such an operation is only allowed if the resources r_1 and r_2 are of the same type (e.g. both have to be teachers). Figure 4 presents an example of this move.

*3.2.5 Resource Move (*RM*)*

The resource r_1 assigned to an event e_1 is replaced by a new resource r_2, randomly selected from the available resources that can be used to attend e_1. Figure 5 presents an example of this move.

Fig. 4 Example of Resource Swap [6].

Left timetable — ES($Geog_3$, His_4):

Mon	Tue	Wed	Thu	Fri
$Math_1$ Smith	Eng_1 Anne	$Math_4$ Smith	$Phis_1$ Laura	Eng_3 Anne
$Math_2$ Smith	Eng_2 Anne	$Math_5$ Smith	$Phis_2$ Laura	Eng_4 Anne
$Math_3$ Smith	$Chem_1$ John	$Geog_3$ Kate	$Span_1$ Mark	Eng_5 Anne
$Geog_1$ Kate	$Chem_2$ John	His_1 Arnald	His_3 Arnald	$Phis_3$ Laura
$Geog_2$ Kate	$Chem_3$ John	His_2 Arnald	His_4 Arnald	

Right timetable:

Mon	Tue	Wed	Thu	Fri
$Math_1$ Smith	Eng_1 Anne	$Math_4$ Smith	$Phis_1$ Laura	Eng_3 Anne
$Math_2$ Smith	Eng_2 Anne	$Math_5$ Smith	$Phis_2$ Laura	Eng_4 Anne
$Math_3$ Smith	$Chem_1$ John	$Geog_3$ Arnald	$Span_1$ Mark	Eng_5 Anne
$Geog_1$ Kate	$Chem_2$ John	His_1 Arnald	His_3 Arnald	$Phis_3$ Laura
$Geog_2$ Kate	$Chem_3$ John	His_2 Arnald	His_4 Kate	

Fig. 5 Example of Resource Move [6].

Left side:

Brandon	
Judith	
Jane	

Mon	Tue	Wed	Thu	Fri
$Math_1$ Smith	Eng_1 Anne	$Math_4$ Smith	$Phis_1$ Laura	Eng_3 Anne
$Math_2$ Smith	Eng_2 Anne	$Math_5$ Smith	$Phis_2$ Laura	Eng_4 Anne
$Math_3$ Smith	$Chem_1$ John	$Geog_3$ Kate	$Span_1$ Mark	Eng_5 Anne
$Geog_1$ Kate	$Chem_2$ John	His_1 Arnald	His_3 Arnald	$Phis_3$ Laura
$Geog_2$ Kate	$Chem_3$ John	His_2 Arnald	His_4 Arnald	

RM($Span_1$, Jane) →

Right side:

Brandon	
Judith	
Mark	

Mon	Tue	Wed	Thu	Fri
$Math_1$ Smith	Eng_1 Anne	$Math_4$ Smith	$Phis_1$ Laura	Eng_3 Anne
$Math_2$ Smith	Eng_2 Anne	$Math_5$ Smith	$Phis_2$ Laura	Eng_4 Anne
$Math_3$ Smith	$Chem_1$ John	$Geog_3$ Kate	$Span_1$ Jane	Eng_5 Anne
$Geog_1$ Kate	$Chem_2$ John	His_1 Arnald	His_3 Arnald	$Phis_3$ Laura
$Geog_2$ Kate	$Chem_3$ John	His_2 Arnald	His_4 Arnald	

*3.2.6 Kempe Move (*KM*)*

Two timeslots t_1 and t_2 are selected. The events assigned to t_1 and t_2 are listed and represented as nodes in a graph. If two nodes (events) n_1 and n_2 in this graph share resources, they are connected with an edge. Edges are created only between nodes assigned in distinct timeslots, thus, the generated graph is a bipartite graph known as conflict graph. Every edge in the conflict graph also has a weight, formed by the cost difference in the objective function assuming the exchange of timeslots between the events in the pair (n_1, n_2). Afterwards, the method looks for the path with the lowest cost in the conflict graph and it makes the exchange of timeslots in the chain. This procedure is similar to that proposed by Tuga *et al.* [28]. Figure 6 presents an example of this move.
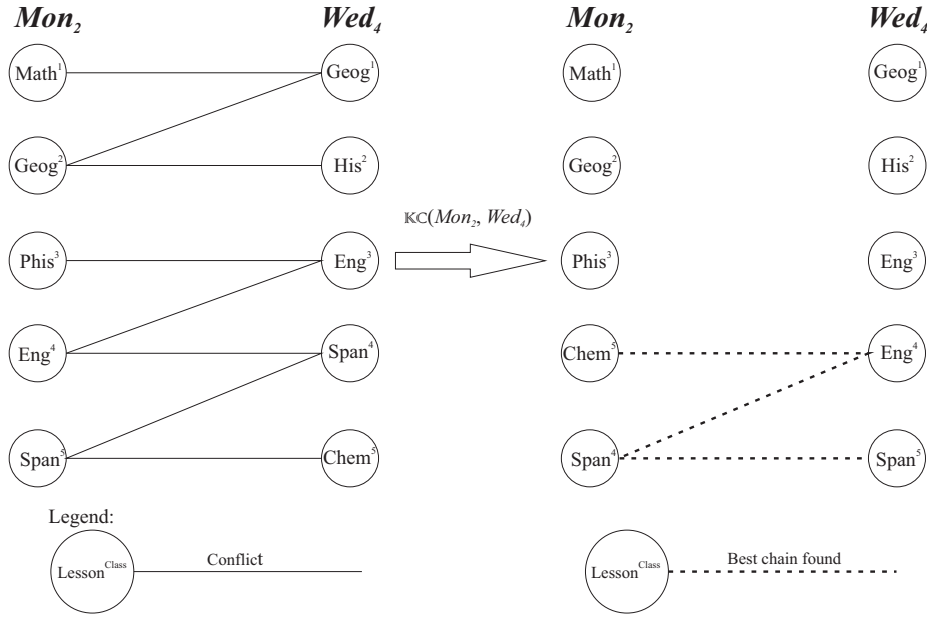
Fig. 6 Example of Kempe Move [6].

3.2.7 Move Selection

The move k in $N(s)$ is randomly selected in order to generate a neighbor. If the instance requires the assignment of resources (i.e. has at least one ASSIGN RESOURCE constraint), the moves are chosen based on the following probabilities: ES = 0.20, EM = 0.38, EBS = 0.10, RS = 0.20, RM = 0.10 and KM = 0.02. Otherwise, the moves RS and RM are not used and the probabilities become: ES = 0.40, EM = 0.38, EBS = 0.20 and KM = 0.02. These values were adjusted based on empirical observation.

3.3 Late Acceptance Hill-Climbing

The Late Acceptance Hill-Climbing metaheuristic was recently proposed by [2]. This algorithm is an adaptation of the classical Hill-Climbing method. It relies on comparing a new candidate solution with the last l-th solution considered in the past, in order to accept or to reject it. Note that the candidate solution may be accepted even if it is worse than the current solution, since it is compared to the solution of l iterations before.

This metaheuristic was created with three goals in mind: to be an one-point search procedure that does not employ an artificial cooling schedule, like Simulated Annealing; to effectively use the information collected during previous iterations of the search, and; to employ a simple acceptance mechanism (i.e. almost as simple as Hill-Climbing) [3].

In this method, a vector $\mathbf{p} = \mathbf{p}_0, \ldots \mathbf{p}_{l-1}$ with costs of previous solutions is stored. Initially this list is filled with the cost of the initial solution s: $\mathbf{p}_k \leftarrow$

$f(s) \ \forall k \in \{0, ..., l-1\}$. At each iteration i, a candidate solution s' is generated. The candidate solution is accepted if its cost is less or equal to the cost stored on the $i \mod l$ position of $\mathbf{p}$. Moreover, if this solution is better than the best solution s^* found so far, a new incumbent solution is stored. Afterwards, the position $v = i \mod l$ of $\mathbf{p}$ is updated: $\mathbf{p}_v \leftarrow f(s')$. This process repeats until a stopping condition is met.

The implementation of the LAHC is illustrated in Algorithm 1. Note that time-out was adopted as the stopping condition for the algorithm. This decision is discussed in Section 4. Some successful examples of application of LAHC can be found in [1, 22, 30].

Algorithm 1: Developed implementation of LAHC

> **Input**: Initial solution s and parameter l.
> **Output**: Best solution s^* found.
> 1 $\mathbf{p}_k \leftarrow f(s) \ \forall k \in \{0, ..., l\text{ - }1\}$;
> 2 $s^* \leftarrow s$;
> 3 $i \leftarrow 0$;
> 4   **while** $elapsedTime < timeout$ **do**
> 5   Generate a random neighbor $s' \in N(s)$;
> 6 $v \leftarrow i \mod l$;
> 7   **if** $f(s') \leq \mathbf{p}_v$ **then**
> 8 $s \leftarrow s'$;
> 9   **if** $f(s) < f(s^*)$ **then**
> 10 $s^* \leftarrow s$;
> 11 $\mathbf{p}_v \leftarrow f(s)$;
> 12 $i \leftarrow i + 1$;
> 13 **return** s^*;

Since it is relatively recent, variations of the LAHC metaheuristic were not extensively explored yet. Therefore, the combination of LAHC with other methods and strategies is an open field for experimentation [3]. In this paper we propose and evaluate computationally two LAHC variants, one of which is a hybrid version including Simulated Annealing.

3.3.1 Stagnation Free LAHC

In late stages of the LAHC execution, it is often very hard to improve the current solution. The algorithm can lead to a list with all l positions occupied with the same cost value, even for large values of l. This behavior can make the LAHC incapable of escaping from local minima, since worse solutions are never accepted. A new variation of the LAHC, so-called Stagnation Free LAHC or simply sf-LAHC, is proposed in this paper in order to handle such situations.

In sf-LAHC method, the algorithm reheats the system when it reaches a stagnation condition. In the proposed implementation, the reheat consists of retrieving the vector of costs from the last time in which one improvement occurred, denoted by $\mathbf{p}'$. It means that various worsening moves may become acceptable after this list update. The algorithm is considered on stagnation when it performs n iterations without improvement. The authors suggest to set n as a function of l, in order to

simplify the parameter tuning process. The Stagnation Free variant of LAHC is presented in Algorithm 2. In the experiments we considered always $n = 1,000 \times l$.

Algorithm 2: sf-LAHC

Input: Initial solution s and parameters l and $mult$.
Output: Best solution s^* found.

1   $n \leftarrow l \times mult$;
2   $\mathbf{p}_k \leftarrow \mathbf{p}'_k \leftarrow f(s) \; \forall k \in \{0, ..., l\text{-}1\}$;
3   $s^* \leftarrow s$;
4   $i \leftarrow 0$;
5   **while** $elapsedTime < timeout$ **do**
6   Generate a random neighbor $s' \in N(s)$;
7      $v \leftarrow i \bmod l$;
8   **if** $f(s') \leq \mathbf{p}_v$ **then**
9         $s \leftarrow s'$;
10   **if** $f(s) < f(s^*)$ **then**
11           $s^* \leftarrow s$;
12           $\mathbf{p}' \leftarrow \mathbf{p}$;
13           $i \leftarrow 0$;
14      $\mathbf{p}_v \leftarrow f(s)$;
15      $i \leftarrow i + 1$;
16   **if** $i = n$ **then**
17        $\mathbf{p} \leftarrow \mathbf{p}'$;
18        $i \leftarrow 0$;
19   **return** s^*;

3.3.2 Simulated Annealing - LAHC

Proposed by [15], the metaheuristic Simulated Annealing (SA) is a probabilistic method based on an analogy to thermodynamics, simulating the cooling of a set of heated atoms. This technique starts its search from any initial solution. The main procedure consists of a loop that randomly generates, at each iteration, one neighbor s' of the current solution s. Movements are probabilistically selected considering a temperature T and the cost variation obtained with the move, Δ.

This algorithm was part of the solvers in all ITC winners [5,16,20]. It also achieved good results in this model of the problem, specially for larger instances. Therefore, it was evaluated in a hybrid approach with the LAHC algorithm. Since Simulated Annealing performance is not strongly affected by the fitness of the initial solution, it has been considered a mixed algorithm, with the Simulated Annealing algorithm being executed in the initial solution, generating a s^* solution, and the LAHC method being executed further, to polish this solution, generating a final solution s^{**}. A combination of Simulated Annealing and sf-LAHC variant of LAHC was also tested. A mixed approach with as-LAHC was not presented because it achieved poor results.

The implementation of Simulated Annealing which is used in this work is described in Algorithm 3. Parameters were set as $\alpha = 0.97$, $T_0 = 1$ and $SAmax = 10,000$. The method $selectMovement()$ chooses a move according to the neighborhood probabilities previously defined.

Algorithm 3: Developed implementation of SA

    Input: $f(.), N(.), \alpha, SAmax, T_0, s, timeout$
    Output: Best solution s^* found.
1 $s^* \leftarrow s; IterT \leftarrow 0; T \leftarrow T_0; reheats \leftarrow 0;$
2 **while** $elapsedTime < timeout$ **do**
3   **while** $IterT < SAmax$ **do**
4 $IterT \leftarrow IterT + 1;$
5 $k \leftarrow selectMovement();$
6   Generate a random neighbor $s^{'} \in N_k(s);$
7 $\Delta = f(s^{'}) - f(s));$
8   **if** $\Delta < 0$ **then**
9 $s \leftarrow s^{'};$
10   **if** $f(s^{'}) < f(s^*)$ **then** $s^* \leftarrow s^{'};$
11   **else**
12   Take $x \in [0, 1];$
13   **if** $x < e^{-\Delta/T}$ **then** $s \leftarrow s^{'};$
14 $T \leftarrow \alpha \times T;$
15 $IterT \leftarrow 0;$
16 **return** $s^*;$

---

4 Computational Experiments

All experiments were executed on an Intel $^{\circledR}$ i5 2.4 Ghz computer, 4GB of RAM, under Ubuntu 11.10 operating system. The software was coded in C++ and compiled with GCC 4.6.1. The obtained results were validated by HSEval validator[2]. The stopping criterion was 1,500 seconds timeout, adjusted according to the ITC2011 provided benchmark.

The results are expressed by the pair x/y, where x stands for the feasibility measure and y for the quality measure. The proposed solver, along with solutions and reports, can be found at GOAL-UFOP website[3]. The interested reader is invited to validate the results.

4.1 Dataset Characterization

The set of instances available from ITC2011 [24] is composed of problems from many countries, ranging from small to large and challenging instances. The main features of the considered instances are presented in Table 1.

4.2 Parameter Setting

One of the key advantages of LAHC is the small number of parameters to be set. Actually, the algorithm has only one parameter, which is the length l of **p** vector. As mentioned by [2], higher values of l make the search more suitable to find better results but also imply a higher processing time. On the other hand, low values of l

---

[2] http://sydney.edu.au/engineering/it/~jeff/hseval.cgi

[3] Code, solutions and reports are available at http://www.goal.ufop.br/softwares/hstt

Table 1 Features of considered instances from ITC2011

Instance	Timeslots	Teachers	Rooms	Classes	Lessons
BrazilInstance2	25	14		6	150
BrazilInstance3	25	16		8	200
BrazilInstance4	25	23		12	300
BrazilInstance6	25	30		14	350
FinlandElementarySchool	35	22	21	291	445
FinlandSecondarySchool2	40	22	21	469	566
Aigio1stHighSchool10-11	35	37		208	532
Italy_Instance4	36	61		38	1101
KosovaInstance1	62	101		63	1912
Kottenpark2003	38	75	41	18	1203
Kottenpark2005A	37	78	42	26	1272
Kottenpark2008	40	81	11	34	1118
Kottenpark2009	38	93	53	48	1301
Woodlands2009	42	40			1353
Spanishschool	35	66	4	21	439
WesternGreeceUniversity3	35	19		6	210
WesternGreeceUniversity4	35	19		12	262
WesternGreeceUniversity5	35	18		6	184

make the search faster but it can lead to poor results. For instance, if one considers $l = 1$, the method performs exactly like the classical Hill-Climbing method.

In this sense, experiments considering many values of l: $l = \{1, 10, 100, 500, 1,000, 5,000, 10,000, 20,000, 50,000\}$ have been executed. The instances *BrazilInstance2*, *ItalyIntance4*, *SpainSchool*, *KosovaInstance* and *NetherlandsKottenpark2009* have been chosen to determinate which value has the better average performance. These instances were chosen since they have different sizes and features. Tables 2 and 3 present the results obtained under the considered configurations.

**Table 2** Experiments considering several values of parameter l on the original LAHC

Size of l	Brazil Instance2	Italy Instance4	Kosova Instance1	Netherlands Kottenpark09	Spain School1
$l = 1$	1 / 67	0 / 391	1 / 114	27 / 5740	0 / 981
$l = 10$	1 / 67	0 / 388	0 / 98	28 / 7735	0 / 561
$l = 100$	1 / 46	0 / 587	0 / 40	28 / 4745	0 / 601
$l = 500$	0 / 74	0 / 82	0 / 66	28 / 4745	0 / 681
$l = 1,000$	0 / 149	0 / 154	0 / 54	23 / 9050	0 /1001
$l = 5,000$	0 / 61	0 / 63	0 / 1430	26 / 13955	0 / 460
$l = 10,000$	0 / 78	0 / 85	162 / 12365	22 / 7165	0 / 686
$l = 20,000$	0 / 58	0 / 109	429 / 22335	25 / 10375	0 / 487
$l = 50,000$	0 / 75	0 / 133	811 / 27396	24 / 52570	0 / 557

The poor performance observed for $l = 1$ was expected, since the algorithms become identical to the original Hill-Climbing method. In general, it is possible to detect two different behaviors:

– for small instances, higher l values imply better performance, since the algorithm capacity of escaping from local optima increases. It can be seen for instance *BrazilInstance2* in Table 3.
– for large instances, the performance of the method increases with l but, after some point, it starts to decrease, because the algorithm does not reach con-

Table 3 Experiments considering several values of parameter l on sf-LAHC

Size of l	Brazil Instance2	Italy Instance4	Kosova Instance1	Netherlands Kottenpark09	Spain School1
$l = 1$	1 / 67	0 / 699	1 / 113	34 / 5750	0 / 991
$l = 10$	0 / 55	0 /598	0 / 99	29 / 7610	0 /601
$l = 100$	0 / 55	1 / 792	0 / 40	29 / 10865	0 / 601
$l = 500$	0 /40	0 / 71	0 / 66	31 / 4995	0 /1027
$l = 1,000$	0 / 61	0 / 53	0 / 54	23 / 12110	0 / 1002
$l = 5,000$	0 / 61	0 / 69	0 / 1285	24 / 8940	0 /4168
$l = 10,000$	0 / 56	0 / 99	161 / 12420	22 / 7495	0 / 4176
$l = 20,000$	0 /41	0 / 128	427 / 21517	28 / 86875	0 / 4163
$l = 50,000$	0 / 16	0 / 9396	814 / 29172	22 / 395120	0 / 4163
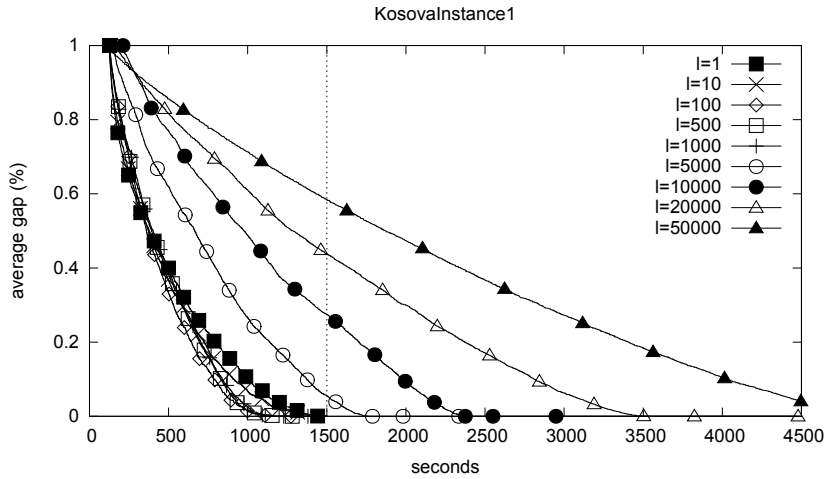
Fig. 7 Behavior of LAHC regarding the l parameter to KosovaInstance1.

vergence before time-out in these cases. The instance *KosovaInstance1*, whose convergence curves are shown in figure reffig:KosovaInstance1, is an example of such a case. From this figure, it is possible to note that both, excessively high or excessively low values of l lead to bad results.

Based on the overall performances of the methods, we fixed $l = 500$ to perform the remaining experiments. This size has been chosen because it has shown to be a good compromise between small and large instances.

4.3 Obtained Results

Table 4 presents the results obtained with the LAHC method and its variants. The results obtained with the KHE engine (initial solution), the ITC2011 winner approach (SA-ILS) and the stand alone Simulated Annealing (SA) are also presented for comparison. The results presented are average values of five runs, with random seeds. The value of "Average ranking" was calculated following the ITC2011 rules: each solution method was ranked between 1 and 5 on each instance (1 being the

best and 5 being the worst), and the average of these ranks was taken. The best results are highlighted in bold.

The Algorithm SA-sf-LAHC has been compared with the other results from ITC2011, since it was the algorithm with better performance among the proposed methods. Such a comparison is shown in Table 5. Decimal values were rounded to the nearest integer. Again, the "Average ranking" is calculated following the ITC2011 rule. The best results are highlighted in bold. In a brief description of ITC2011 finalists, GOAL team [5] developed a SA-ILS hybrid local search approach; HySTT team [10] developed a method based on Hyper-heuristics; Lectio team [27] used an Adaptive Large Neighborhood Search and HTF team [25] used an Evolutionary Algorithm.

4.4 Discussion of Results

In some instances, even the production of feasible solutions is complicated, specially when most constraints are set as hard ones. The LAHC method and its variants were able to find 12 feasible solutions out of the 18 instances in the considered dataset, one more than the ITC2011 winner. In Table 4, it is possible to see that the LAHC algorithm and its variants outperformed the SA-ILS solver.

When compared to the other methods, the standalone Simulated Annealing worked better in large instances than in small ones. Surprisingly, the SA algorithm had better performance than the hybrid SA-ILS approach. When it is compared to the original LAHC, it is possible to note that the LAHC was slightly better than the SA. This is an interesting result, since LAHC is a new metaheuristic and it is still open for improvements. Besides, the Simulated Annealing is known as a good algorithm for dealing with scheduling problems, what makes the observed result a good achievement.

The Stagnation Free version of LAHC obtained good results, outperforming its original version in several instances. This could be noted specially in small instances, in which sf-LAHC can keep some improvement until the time-out is reached instead of the original LAHC, which probably got stuck at a local optima. Finally, it is important to highlight the remarkable performance observed for the combination of LAHC and SA proposed in this work (SA-sf-LAHC). This heuristic obtained the best results and, compared to the finalist results (see Table 5), it is possible to conclude that it would win the competition by a large margin: it reached the best result in 14 out of 18 instances, leading to an overall ranking of 1.42. A two tail Welchs T-test, comparing GOAL and SA-sf-LAHC rankings, reinforced the assumption of SA-sf-LAHC superiority: it has obtained a p-value of 8.0254e-06, which widely supports the rejection of the null hypothesis (equivalent algorithms) under the confidence level of 95%.

5 Concluding Remarks

This work presented an application of the Late Acceptance Hill-Climbing algorithm to the High School Timetabling Problem model proposed in the ITC2011. In addition, some variants of the LAHC method were proposed and evaluated computationally.

Table 4: Comparsion between LAHC based approaches and SA based methods

Instance	KHE	SA-ILS	SA	LAHC	sf-LAHC	SA-LAHC	SA-sf-LAHC
BrazilInstance2	4 / 90	1.0 / 63.9	0.0 / 107.6	0.0 / 78.6	**0.0** / **52.8**	0.0 / 102.4	0.0 / 78.0
BrazilInstance3	3 / 240	**0.0** / **127.8**	0.0 / 170.6	0.0 / 145.6	0.0 / 137.0	0.0 / 174.0	0.0 / 160.2
BrazilInstance4	39 / 144	17.2 / 99.6	**2.0** / **167.4**	8.2 / 121.4	6.0 / 112.8	2.4 / 165.6	2.4 / 164.2
BrazilInstance6	11 / 291	4.0 / 223.5	0.0 / 332.2	1.4 / 204.8	0.6 / 166.4	0.0 / 252.6	**0.0** / **221.0**
FinlandElementarySchool	9 / 30	0.0 / 4.0	0.0 / 10.0	0.0 / 3.8	**0.0** / **3.6**	0.0 / 3.8	0.0 / 3.8
FinlandSecondarySchool2	2 / 1821	0.0 / 0.4	0.0 / 1036.4	0.0 / 0.4	**0.0** / **0.2**	**0.0** / **0.2**	0.0 / 0.4
Aigio1stHighSchool10-11	14 / 757	**0.0** / **15.3**	0.0 / 397.4	0.8 / 11.4	0.8 / 11.4	0.0 / 17.8	0.0 / 20.8
Italy_Instance4	39 / 21238	0.0 / 658.4	0.0 / 13979.0	0.0 / 224.8	**0.0** / **199.4**	0.0 / 366.4	0.0 / 302.2
KosovaInstance1	1333 / 566	14.0 / 6934.4	3.0 / 5837.8	14.2 / 1504.8	**0.0** / **137.0**	4.6 / 5238.6	6.3 / 6383.8
Kottenpark2003	3 / 78440	0.6 / 90195.8	0.4 / 90052.4	1.6 / 9711.0	1.6 / 9136.8	**0.0** / **69608.2**	0.4 / 89132.2
Kottenpark2005A	35 / 23677	33.9 / 27480.4	30.0 / 33967.0	33.0 / 18671.0	32.8 / 18891.8	30.2 / 33310.4	**30.2** / **33169.6**
Kottenpark2008	63 / 140083	25.7 / 31403.7	**10.0** / **138993.8**	15.2 / 23855.0	14.4 / 23067.2	10.8 / 57476.6	10.8 / 59939.2
Kottenpark2009	55 / 211095	36.6 / 154998.5	**24.6** / **432784.0**	28.0 / 9192.0	28.8 / 8363.0	25.4 / 112948.0	25.8 / 112335.0
Woodlands2009	19 / 0	2.0 / 15.8	2.0 / 223.8	2.0 / 13.2	**2.0** / **12.0**	2.0 / 12.4	**2.0** / **12.0**
Spanish school	1 / 4103	0.0 / 865.2	0.0 / 7077.6	**0.0** / **846.0**	0.0 / 920.4	0.0 / 998.2	0.0 / 856.8
WesternGreeceUniversity3	0 / 30	0.0 / 5.6	0.0 / 28.4	**0.0** / **5.0**	**0.0** / **5.0**	0.0 / 5.4	0.0 / 5.2
WesternGreeceUniversity4	0 / 41	0.0 / 7.4	0.0 / 39.4	0.0 / 9.2	0.0 / 6.8	0.0 / 7.6	**0.0** / **6.6**
WesternGreeceUniversity5	17 / 44	**0.0** / **0.0**	0.0 / 56.0	**0.0** / **0.0**	**0.0** / **0.0**	**0.0** / **0.0**	**0.0** / **0.0**
Average ranking	-	4.50	4.11	4.00	2.83	3.03	**2.53**

Table 5 Comparison of results between SA-sf-LAHC and ITC2011 finalists.

Instance	SA-sf-LAHC	GOAL [5]	HySTT [10]	Lectio [27]	HFT [25]
BrazilInstance2	0 / 78	1 / 62	1 / 77	**0 / 38**	6 / 190
BrazilInstance3	0 / 160	0 / 124	**0 / 118**	0 / 152	30 / 283
BrazilInstance4	**2 / 164**	17 / 98	4 / 231	2 / 199	67 / 237
BrazilInstance6	**0 / 221**	4 / 227	3 / 269	0 / 230	23 / 390
FinlandElementarySchool	0 / 4	0 / 4	1 / 4	**0 / 3**	30 / 73
FinlandSecondarySchool2	**0 / 0**	0 / 1	0 / 23	0 / 34	31 / 1628
Aigio1stHighSchool10-11	0 / 21	**0 / 13**	2 / 470	0 / 1062	50 / 3165
Italy_Instance4	**0 / 302**	0 / 454	0 / 6926	0 / 651	263 / 6379
KosovaInstance1	**6 / 6384**	59 / 9864	1103 / 14890	275 / 7141	989 / 39670
Kottenpark2003	0 / 89132	0 / 90928	1 / 56462	50 / 69773	209 / 84115
Kottenpark2005A	**30 / 33170**	31 / 32108	32 / 30445	350 / 91566	403 / 46373
Kottenpark2008	**11 / 59939**	13 / 33111	141 / 89350	209 / 98663	-
Kottenpark2009	**26 / 112335**	28 / 12032	38 / 93269	128 / 93634	345 / 99999
Woodlands2009	**2 / 12**	2 / 14	2 / 70	1 / 107	62 / 338
Spanish school	**0 / 857**	0 / 894	0 / 1668	0 / 2720	65 / 13653
WesternGreeceUniversity3	**0 / 5**	0 / 6	0 / 11	30 / 2	15 / 190
WesternGreeceUniversity4	**0 / 7**	**0 / 7**	0 / 21	36 / 95	237 / 281
WesternGreeceUniversity5	**0 / 0**	**0 / 0**	0 / 4	4 / 19	11 / 158
Average ranking	**1.42**	2.19	3.22	3.22	4.94

The LAHC algorithm obtained good results. It was able to outperform the standalone Simulated Annealing approach and the ITC2011 winner approach, a SA-ILS method. The LAHC variants proposed in this paper also reached promising results. The Stagnation Free LAHC (sf-LAHC) was able to outperform its original version. The combinations of LAHC and sf-LAHC with Simulated Annealing were tested, and the mixed SA-sf-LAHC algorithm achieved the best results to this problem up to now. One great feature of LAHC algorithm is its simplicity: it is very easy to implement and it relies only on one parameter to be tuned.

Some possible future extensions of this work are (*i*) to develop and to evaluate other variations of LAHC as suggested by [2]; (*ii*) to implement and to evaluate other neighborhood moves, and; (*iii*) to develop a graphical user interface to allow the use of the solver by schools and universities.

References

1. Abuhamdah, A.: Experimental result of late acceptance randomized descent algorithm for solving course timetabling problems. In: IJCSNS- International Journal of Computer Science and Network Security, Vol. 10 No. 1, January (2010)
2. Burke, E.K., Bykov, Y.: A Late Acceptance Strategy in Hill-Climbing for Exam Timetabling Problems. In: PATAT '08 Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (2008)
3. Burke, E.K., Bykov, Y.: The late acceptance hill-climbing heuristic. Tech. Rep. CSM-192, Department of Computing Science and Mathematics, University of Stirling (2012)
4. Dorneles, Á.P., de Araújo, O.C., Buriol, L.S.: A fix-and-optimize heuristic for the high school timetabling problem. Computers & Operations Research **52**, 29–38 (2014)
5. Fonseca, G., Santos, H., Toffolo, T., Brito, S., Souza, M.: A SA-ILS approach for the High School Timetabling Problem. PATAT '12 Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling (2012)
6. Fonseca, G.H.G., Santos, H.G., Toffolo, T.n.M., Brito, S.S., Souza, M.J.F.: GOAL solver: a hybrid local search based solver for high school timetabling. Annals of Operations Research pp. 1–21 (2014). DOI 10.1007/s10479-014-1685-4
7. Garey, M.R., Jonhson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco, CA, USA (1979)

8. de Haan, P., Landman, R., Post, G., Ruizenaar, H.: A case study for timetabling in a dutch secondary school. In: Lecture notes in computer science: VI Practice and theory of automated timetabling. Berlin : Springer, pp. 3867 : 267–279 (2007)
9. IDSIA: International Timetabling Competition 2002 (2012). Available at `http://www.idsia.ch/Files/ttcomp2002/`, Accessed in December / 2012
10. Kheiri, A., Ozcan, E., Parkes, A.J.: Hysst: Hyper-heuristic search strategies and timetabling. In: Proceedings of the ninth international conference on the practice and theory of automated timetabling (PATAT 2012), pp. 497–499 (2012)
11. Kingston, J.: KHE14 an algorithm for high school timetabling. 10th International Conference on the Practice and Theory of Automated Timetabling pp. 26–29 (2014)
12. Kingston, J.H.: A tiling algorithm for high school timetabling. In: Lecture notes in computer science: V Practice and theory of automated timetabling. Berlin: Springer, pp. 3616 : 208–225 (2005)
13. Kingston, J.H.: A software library for school timetabling (2012). Available at `http://sydney.edu.au/engineering/it/~jeff/khe/`, May 2012
14. Kingston, J.H.: A software library for school timetabling (2012). Available at `http://sydney.edu.au/engineering/it/~jeff/khe/`, Accessed in December / 2012
15. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**, 671–680 (1983)
16. Kostuch, P.: The university course timetabling problem with a three-phase approach. In: Proceedings of the 5th international conference on Practice and Theory of Automated Timetabling, PATAT'04, pp. 109–125. Springer-Verlag, Berlin, Heidelberg (2005). DOI 10.1007/11593577_7
17. Kristiansen, S., Srensen, M., Stidsen, T.: Integer programming for the generalized high school timetabling problem. Journal of Scheduling pp. 1–16 (2014). DOI 10.1007/s10951-014-0405-x
18. McCollum, B.: International Timetabling Competition 2007 (2012). Available at `http://www.cs.qub.ac.uk/itc2007/`, Accessed in December / 2012
19. Moura, A.V., Scaraficci, R.A.: A grasp strategy for a more constrained school timetabling problem. International Journal of Operational Research **7**(2), 152–170 (2010)
20. Muller, T.: ITC2007 solver description: a hybrid approach. Annals OR **172**(1), 429–446 (2009). URL `http://dblp.uni-trier.de/db/journals/anor/anor172.html#Muller09`
21. Nurmi, K., Kyngas, J.: A framework for school timetabling problem. In: Proceedings of the 3rd multidisciplinary international scheduling conference: theory and applications, Paris, pp. 386–393 (2007)
22. Özcan, E., Bykov, Y., Birben, M., Burke, E.K.: Examination timetabling using late acceptance hyper-heuristics. In: Proceedings of the Eleventh conference on Congress on Evolutionary Computation, CEC'09, pp. 997–1004. IEEE Press, Piscataway, NJ, USA (2009). URL `http://dl.acm.org/citation.cfm?id=1689599.1689731`
23. Pillay, N.: A survey of school timetabling research. Annals of Operations Research pp. 1–33 (2013)
24. Post, G., Kingston, J., Ahmadi, S., Daskalaki, S., Gogos, C., Kyngas, J., Nurmi, C., Musliu, N., Pillay, N., Santos, H., Schaerf, A.: XHSTT: an XML archive for high school timetabling problems in different countries. Annals of Operations Research **218**(1), 295–301 (2014)
25. Romrs, J., Homberger, J.: An evolutionary algorithm for high school timetabling. PATAT '12 Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling (2012)
26. Santos, H.G., Uchoa, E., Ochi, L.S., Maculan, N.: Strong bounds with cut and column generation for class-teacher timetabling. Annals OR **194**(1), 399–412 (2012)
27. Srensen, M., Kristiansen, S., Stidsen, T.: International Timetabling Competition 2011: An Adaptive Large Neighborhood Search algorithm, pp. 489–492 (2012)
28. Tuga, M., Berretta, R., Mendes, A.: A hybrid simulated annealing with kempe chain neighborhood for the university timetabling problem. In: Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference on, pp. 400–405. IEEE (2007)
29. Valourix, C., Housos, E.: Constraint programming approach for school timetabling. In: Computers & Operations Research, pp. 30 : 1555–1572 (2003)
30. Verstichel, J., Vanden Berghe, G.: A late acceptance algorithm for the lock scheduling problem. In: S. Voss, J. Pahl, S. Schwarze (eds.) Logistik Management, pp. 457–478. Springer (2009)
31. Wright, M.: School timetabling using heuristic search. In: Journal of Operational Research Society, pp. 47 : 347–357 (1996)