Constraint Programming for High School Timetabling: A Scheduling-Based Model with Hot Starts

Emir Demirović and Peter J. Stuckey

School of Computing and Information Systems, University of Melbourne, Australia
(edemirovic,pstuckey)@unimelb.edu.au

Abstract. High School Timetabling (HSTT) is a well-known and wide-spread problem. It consists of coordinating resources (e.g. teachers, rooms), times, and events (e.g. classes) with respect to a variety of constraints. In this paper, we study the applicability of constraint programming (CP) for high school timetabling. We formulate a novel CP model for HSTT using a scheduling-based point of view. We show that a drastic improvement in performance over the baseline CP model can be achieved by including solution-based phase saving, which directs the CP solver to first search in close proximity to the best solution found, and our hot start approach, where we use existing heuristic methods to produce a starting point for the CP solver. The experiments demonstrate that our approach outperforms the IP and maxSAT complete methods and provides competitive results when compared to dedicated heuristic solvers.

Keywords: constraint programming, timetabling, scheduling, modeling, hot start, warm start, local search, phase saving

1 Introduction

The problem of high school timetabling (HSTT) is to coordinate resources (e.g. rooms, teachers, students) with times to fulfill certain goals (e.g. scheduling classes). Every school requires some form of HSTT, making it a wide-spread problem. In a more general sense, timetabling can be found in airports, transportation, and the like. The quality of the timetables is an important issue, since they have a direct impact on the educational system, satisfaction of students and staff, and other matters. Every timetable affects hundreds of students and teachers for prolonged periods of time, as each timetable is used for at least a semester, making HSTT an extremely important and responsible task. However, constructing timetables by hand can be time consuming, difficult, error prone, and in some cases practically impossible. Thus, developing algorithms to produce the best timetables automatically is of utmost importance.

There has been significant research tackling HSTT. However, given that there are many educational systems, each differing in their own ways, much of this research was done in isolation targeting only a particular set of rules. It was difficult to compare developed algorithms due to the differences, even though the problems shared similarities. This motivated researchers to develop a general high school timetabling formulation, named XHSTT [24, 25], that allows a variety of education system requirements to be expressed. With the new formulation, researchers now have common ground for fair algorithmic comparison. In 2011, the International Timetabling Competition was dedicated to HSTT and endorsed the said formulation, encouraging further research in the direction of XHSTT. We consider this formulation in our work.

Historically, incomplete algorithms were the most prominent for XHSTT (e.g. [12, 11, 15]. Recently, complete methods based on integer programming [16] and maximum Boolean satisfiability (maxSAT) [8] have proven to be effective. Their development was essential for the emergence of large neighborhood search algorithms, which combine domain-specific heuristics with complete solving [27, 7]. These methods are currently among the most successful ones for XHSTT.

As complete methods play a vital role, it is natural to ask if unexplored complete paradigms could bring more to XHSTT than those currently in use. This is precisely what we study in this work: the applicability of constraint programming for XHSTT. We provide a novel CP model that views XHSTT as a scheduling problem, in contrast to the conventional Boolean formulations. Such a model allows us to exploit high-level global constraints, providing an elegant and possibly more efficient solution. However, to match the state-of-the-art, that is not enough. Indeed, our experimentation demonstrates that a standard CP approach is not competitive. Therefore, further development was required. The first improvement came from employing *solution-based phase saving* [4], an existing technique in maxSAT [1] but not well-known in CP, that directs the CP solver to search in close proximity to the best solution found so far before expanding further. The second was born out of the realization that the community has built sophisticated heuristic algorithms, which can be communicated to the CP solver (so called *hot start*). While usual hot starts in CP would only provide an initial bound to the problem, when combined with the solution phase saving, such an approach offers more: in addition to the bound, it suggests to the solver a region in the search space that is likely to contain high quality solutions. The end result is a complete algorithm that outperforms integer programming and maxSAT methods, while being competitive with dedicated heuristic solvers.

The paper is organized as follows. In the next section we briefly describe the general high school timetabling (XHSTT) problem. In Section 3, we provide an overview of the state-of-the-art for XHSTT. The scheduling-based model is given in Section 4. Our hot start approach is discussed in Section 5, along with solution-based phase saving. Experiments are given in Section 6. We conclude in the last section.

2 Problem Description

In our research we consider the general formulation of the High School Timetabling problem (called XHSTT), as described in [25]. This formulation is general enough to be able to model education systems from different countries and was endorsed by the International Timetabling Competition 2011.

The general High School Timetabling formulation specifies three main entities: times, resources and events. Times refer to discrete time units which are available, such as Monday 9:00-10:00, Monday 10:00-11:00, for example. Resources correspond to available rooms, teachers, students, and others. The main entities are the events, which in order to take place require certain times and resources. An event could be a Mathematics class, which requires a math teacher (which needs to be determined) and a specific student group (both the teacher and the student group are considered resources) and two times. Events are to be scheduled into one or more *solution events* or *subevents*. For example, a Mathematics class with total duration of four hours can be split into two subevents with duration two, but can be scheduled as one subevent with duration four (constraints may impose further constraints on the durations of subevents).

The aim of XHSTT is to find a schedule by assigning times and resources to events such that that all hard constraints are satisfied and the sum of soft constraint violations is minimized.

Constraints impose limits on what kind of assignments are legal. These may constrain that a teacher can teach no more than five classes per day, that younger students should attend more demanding subjects (e.g. Mathematics) in the morning, to name a few examples. It is important to differentiate between hard constraints and soft constraints. The former are important constraints which are given precedence over the latter, in the sense that any single violation of any hard constraint is more important than all of the soft constraints combined. Thus, one aims to satisfy as many hard constraints as possible, and then optimize for the soft constraints. In this sense, "hard constraints" are not, in fact, hard as in the usual meaning used in combinatorial optimisation. Each constraint has a nonnegative cost function associated with it, which penalizes assignments that violate it. The goal is to first minimize the hard constraint costs and then minimize the soft constraint costs. In the general formulation, any constraint may be declared hard or soft and no constraint is predefined as such, but rather left as a modeling option based on the concrete timetabling needs. Each constraint has several parameters, such as to which events or resources it applies and to what extent (e.g. how many idles times are acceptable during the week), and other features, allowing great flexibility.

We now give an informal overview of all the constraints in XHSTT (as given in [25]). For more details regarding the problem formulation, see [24, 25]. There is a total of 16 constraints (plus preassignments of times or resources to events, which are not listed).

Constraints related to events:

1. *Assigned Time* - assign the specified amount of times to specified events.
2. *Preferred Times* - when assigning times to events, specified times are preferred over others.
3. *Link Events* - specified events must take place at the same time.
4. *Spread Events* - specified events must be spread out during the week.

5. *Distribute Split Events* - limits the number of subevents that may take a particular duration for a given event.
6. *Split Events* - limits the minimum and maximum durations and number of subevents for a given event. Together with *Distribute Split Events* this gives fine control on the subevents.
7. *Order Events* - specified events must be scheduled one after the other with nonnegative time-lag in between them.
8. *Avoid Split Assignments* - for all subevents derived from an event, assign the same resources.

Constraints related to resources:

1. *Assigned Resource* - assign specified resources to specified events.
2. *Avoid Clashes* - specified resources cannot be used by two or more subevents at the same time.
3. *Preferred Resources* - when assigning resources to events, specified resources are preferred over others.
4. *Avoid Unavailable Times* - specified resources cannot be used at specified times.
5. *Limit Workload* - specified resources must have their workload lie between given values.
6. *Limit Busy Times* - the amount of times when a resource is being used within specified time groups should lie between given values.
7. *Cluster Busy Times* - specified resources' activities must all take place within a minimum and maximum amount of time groups.
8. *Limit Idle Times* - specified resources must have their number of idle times lie between given values within specified time groups.

3 Related Work

For HSTT, both heuristic and complete methods have been proposed. Heuristic methods were historically the dominating approach, as they are able to provide good solutions in a reasonable amount of time even when dealing with large instances, albeit not being able to obtain or prove optimality. Recently developed complete methods [5, 26, 30, 31] are successful in obtaining good results and proving bounds but require significantly more time (days or weeks).

The best algorithms from the International Timetabling Competition 2011 (ITC 2011) were incomplete algorithms. The winner was the group GOAL, followed by Lectio and HySST. In GOAL, an initial solution is generated, which is further improved by using Simulated Annealing and Iterated Local Search, using seven different neighborhoods [12]. Lectio uses an Adaptive Large Neighborhood Search [29] with nine insertion methods based on the greedy regret heuristics [32] and fourteen removal methods. HySST uses a Hyper-Heuristic Search [14].

After the competition, the winning team of ITC 2011 developed several new Variable Neighborhood Search (VNS) approaches [11]. All of the VNS approaches have a common search pattern: from one of the available neighborhoods, a random solution is chosen, after which a descent method is applied and the solution is accepted if it is better than the previous best one. Each iteration starts from the best solution. The Skewed Variable Neighborhood was the most successful VSN approach, in which a relaxed rule is used to accept the new solution based on its cost and its distance from the best solution.

Kingston [15] introduced an efficient heuristic algorithm which directly focuses on repairing *defects* (violations of constraints). Defects are examined individually and specialized procedures are developed for most constraints to repair them. KHE14 provides high quality solutions in a low amount of time, but does not necessarily outperform other methods with respect to quality of solution.

Two complete methods have been studied: IP- [16] and maxSAT-based [8] approaches. Neither method strictly dominates the other, as their relative performance depends on the instance. Both models use Boolean variables to explicitly encode if an event is taking place at a particular time, with the main differences being in the expressiveness of the formalism to define the constraints. Overall, the maxSAT approach provides better results but is limited to problems where resources are preassigned to events. A Satisfiability Modulo Theories (SMT) approach has also been investigated in [6] but cannot handle XHSTT instances efficiently. The bitvector-SMT model rather serves as an efficient way of representing XHSTT for local search algorithms, as all constraint costs can be computed using simple bitvector operations.

The developed complete methods were used in large neighborhood search algorithms: IP-based [30] and maxSAT-based [7]. These approaches offer improvements over their complete counterparts when given limited time.

Additionally, several IP- [10, 26, 28, 33] and CP-based [13, 34, 19] techniques have been introduced for related HSTT problems. There are several notable differences in our work compared to the other CP approaches [34, 19]: our modelling is more general as it applies to XHSTT and we demonstrate how to use generic tools to solve XHSTT without intertwining other techniques other than an initial solution procedure, which is clearly decoupled from the rest of our method.

4 Modeling

The key elements of XHSTT are a set of events E, a set of resources R and a set of times T which we shall regard as integers $T = \{0, 1, 2, ...|T| - 1\}$. We considered the restricted form of the problem where resources used by each event are predefined. The majority of the benchmarks from the repository of the International Timetabling Competition fall into this category. Earlier models for the general high school timetabling problem used explicit representations for each pair of events and time slots, indicating whether the event is taking place at that particular time. In contrast, we use a scheduling-based modeling approach, where each subevent is linked to two variables: the starting time and duration variable. As a result, we are able to exploit the `disjunctive` and `regular` global constraints in our model. We now describe the decision variables and then proceed with the modeling of each constraint.

4.1 Decision Variables

Each event $e \in E$ has a maximum total duration $D(e)$. For each event e, we create $D(e)$ subevents, numbered from $0..D(e)-1$. Every subevent is associated with two variables indicating its starting time and duration. We label these as $start(e,i)$ and $dur(e,i)$. The special start time $UN = |T|$ is use to denote that a subevent is not used, in which case its corresponding duration is zero. Constraints may impose restrictions on the amount and duration of the subevents.

Example 1. Let e be an event of duration 3. A total of six variables are allocated: three pairs of starting time and duration variables. Let the assignments be as shown in Table 1(a):

i	$start(e,i)$	$dur(e,i)$
0	5	2
1	10	1
2	UN	0

(a)

i	$start(e,i)$	$dur(e,i)$
0	5	2
1	UN	0
2	10	1

(b)

Table 1. Decision variable assignments for event e of total duration 3: (a) satisfyies the symmetry breaking constraints, while (b) does not.

The assignments state that for event e, two subevents of durations 2 and 1 are scheduled at starting times 5 and 10, respectively. □

The following constraints formally define the decision variables for each event e and its subevent i:

$$\begin{aligned}
&start(e,i) \in T \cup \{UN\}, \\
&dur(e,i) \in \{0,1,...,D(e)\}, \\
&start(e,i) + dur(e,i) \leq |T|, \\
&start(e,i) = UN \Leftrightarrow dur(e,i) = 0
\end{aligned} \tag{1}$$

Symmetry breaking constraints forbid equivalent solutions. These order the subevents by decreasing duration as the primary criteria, and then by start time:

$$\begin{aligned}
&dur(e,i-1) \geq dur(e,i), \\
&dur(e,i-1) = dur(e,i) \Rightarrow start(e,i-1) \leq start(e,i)
\end{aligned} \tag{2}$$

Example 2. Consider the same setting as in the previous example, but with the following assignments shown in Table 1(b). The constraints in Equations 1 are satified, but symmetry breaking constraints (Equations 2) require that subevents are ordered by duration. Furthermore, if $D(e) = 5$ and the $dur(e,0)+dur(e,1)+$

$dur(e, 2) = 5$, and $dur(e, 1) = 2$ and $start(e, 1) = 0$ then the assignment would again be invalid, as in case of ties in duration, subevents are to be sorted by increasing time assignments.

Symmetry breaking constraints, apart from preventing equivalent solutions, contribute towards simpler constraints (e.g. see *Split Events* constraint). We note that non-overlapping of subevents is left to the *Avoid Clashes* constraints.

4.2 Additional notation

Let $EV(r) \subseteq E$ denote the set of events that require resource r. We introduce auxiliary Boolean variables $busy(r, t)$ to indicate that one of the events $e \in EV(r)$ that require $r \in R$ are taking place at time t. This dual viewpoint on the decisions is required for the *Limit Busy Times* and *Cluster Busy Times* constraints. The mapping between the two viewpoints is managed by a straightforward decomposition.

$$busy(r, t) \Leftrightarrow \exists_{e \in EV(r), i \in \{0..D(e)-1\}} start(e, i) \leq t \wedge start(e, i) + dur(e, i) > t$$

A time group $TG \subseteq T$ is a fixed set of times. An event group $EG \subseteq R$ is a fixed set of events.

4.3 Objectives and Constraints

In the problem formulation, constraints are not predefined as hard or soft, but this option is left to the modelers as appropriate for the particular school under consideration. For simplicity of the presentation, we present in the following text the constraints as if they are given as hard constraints. Soft versions are created by reifying the hard formulation. Note that this may mean that global constraints in the soft versions make use of a decomposition instead, explicitly encoding the global constraint using a set of smaller and simpler constraints.

We define the predicate *within* to ease further notation:

$$within(x, l, u) = (x \geq l \wedge x \leq u) \tag{3}$$

Soft constraints are similar to their hard counterparts, but instead penalize each constraint violation instead of forbidding it entirely. For example, soft constraints commonly state that a certain value has to be within given bounds. In the soft case, the deviation is penalized based on the linear or quadratic distance (specified by the constraint) of the said value from the imposed bounds. In the linear case, the violation is calculated as

$$within\_viol(x, l, u) = \max\{0, x - u, l - x\}. \tag{4}$$

Hard constraints essentially have large weights compared to soft constraints. In our model, the hard XHSTT constraints are posed as hard CP constraints, meaning the infeasibility value is not tracked. This modelling choice allows us to drastically simplify the modelling, which leads to an increase in performance for CP but also allows advanced CP techniques to be exploited through global constraints. We now proceed with the modeling of each constraint.

Assigned Times: Events must be assigned their prescribed number of times.

$$\sum_{i\in\{0,1,...,D(e)-1\}} dur(e,i) = D(e) \tag{5}$$

Preferred Times: Subevents of specified event e may start only within the stated time group TG_e. If an optional parameter d is given, the constraint only applies to subevents of that particular duration.

$$\forall i \in \{0,1,...,D(e)-1\} \qquad start(e,i) \neq UN \Rightarrow start(e,i) \in TG_e \tag{6}$$

Link Events: Certain events must simultaneously take place. Let EG be an event group of linked events, all of which have the same total duration TD, i.e. $\forall e \in EG, D(e) = TD$. We make use of the global constraint `all_equal` [2], which enforces that its input variables must be assigned the same values.

$$\begin{aligned}&\forall i \in \{0,1,...,TD-1\}\\ &\quad \texttt{all\_equal}(\big[start(e,i) \mid e \in EG\big]),\\ &\quad \texttt{all\_equal}(\big[dur(e,i) \mid e \in EG\big])\end{aligned} \tag{7}$$

Spread Events: Limits the number of starting times events from specified event groups may have in given time groups. Event and time groups are sets of events and contiguous times, respectively. Let EG and TG denote one such pair with the limits $mine..maxe$.

$$\begin{aligned}&z = \textstyle\sum_{e\in EG, i\in\{0,..,D(e)-1\}} within(start(e,i), \min(TG), \max(TG)) \wedge\\ &within(z, mine, maxe)\end{aligned} \tag{8}$$

Distribute Split Events: Limits the number of subevents of e a given duration d to be in the range $minds_e..maxds_e$.

$$a = \textstyle\sum_{i\in\{0,1,...,D(e)-1\}}(dur(e,i) = d) \ \wedge \ within(a, minds_e, maxds_e) \tag{9}$$

Split Events: Regulates the number of subevents of e between $mins_e..maxs_e$ and the duration of subevents of e between $mind_e..maxd_e$

$$\forall i \in \{0,..,mins_e - 1\}: \qquad start(e,i) \neq UN \ \wedge dur(e,i) \neq 0 \tag{10}$$

$$\forall i \in \{maxs_e,..,D(e)-1\}: \qquad start(e,i) = UN \wedge dur(e,i) = 0 \tag{11}$$

$$dur(e,i) \leq maxd_e \ \wedge \ dur(e,i) \neq 0 \Rightarrow dur(e,i) \geq mind_e \tag{12}$$

Order Events For a given pair of events (e_1, e_2), the constraint imposes that e_1 must take place before e_2. In addition, there must be a minimum $mino_{ep}$ and maximum $maxo_{ep}$ units of time apart.

$$\begin{aligned} & oe = min\{start(e_2, i) \mid i \in \{0..D(e_2) - 1\}\} - \\ & \quad max\{start(e_1, i) + dur(e_1, i) \mid i \in \{0..D(e_1) - 1\}\} \wedge \\ & within(eo, min_{ep}, max_{ep}) \end{aligned} \tag{13}$$

Avoid Clashes: A resource can be used by at most one event at any given time. Here we make use of the global constraint `disjunctive` [9, 17], which takes two arrays s and d of variables as input, where $s[i]$ and $d[i]$ represent the starting time and duration of task i, and enforces no overlap between the tasks.

$$\begin{aligned} & \texttt{disjunctive}(\\ & \quad \big[start(e, i) \mid e \in EV(r), i \in \{0..D(e) - 1\}\big], \\ & \quad \big[dur(e, i) \mid e \in EV(r), i \in \{0..D(e) - 1\}\big])) \end{aligned} \tag{14}$$

Avoid Unavailable Times: Resources cannot be used at specified times. For each resource r and forbidden time t, this is encoded by creating a dummy event that requires r and is fixed at time t with duration 1. The newly created events are added to $events(r)$ and will be considered in the *Avoid Clashes* constraint (above). For the soft version the duration of these dummy events is 0..1 and the constraint is violated if the duration used in 0.

Limit Busy Times: If a resource r is busy in a time group TG, its number of busy times within the time group is restricted to be in $minb_r..maxb_r$.

$$\begin{aligned} & c = \textstyle\sum_{t \in TG}(busy(r, t)) \qquad \wedge \\ & c \neq 0 \Rightarrow within(c, minb_r, maxb_r) \end{aligned} \tag{15}$$

Cluster Busy Times: A resource is busy in a time group TG if it is busy at least one time int the time group. This constraint gives a set of time groups $\mathbf{TG}$ and limits the total number of time groups $TG \in \mathbf{TG}$ that the resource may be busy to the range $mint..maxt$. For example, a teacher must finish his or her work within three days.

$$\begin{aligned} & b = \textstyle\sum_{TG \in \mathbf{TG}}(\exists_{t \in TG} busy(r, t)) \qquad \wedge \\ & within(b, mint, maxt) \end{aligned} \tag{16}$$

Limit Idle Times: Some resources must not have idle times in their schedule. An idle time occurs at time t within contiguous time group TG if the resource is busy at times from TG before and after t and not busy at time t. This is encoded using the `regular` global constraint [22], which takes as input a sequence of variables that must satisfy the automaton constraint, and a deterministic finite automata A defined by a set of states Q, a set of values of the variable sequence S, a transition function which given a state and value defines the next state to reach, an initial state, and a set of final states. It constrains that the transition sequence defined by the sequence variables starting from the start state leads to a valid final state.

The automata has four states $Q = \{q_0, q_1, q_2, f\}$ with the interpretation: q_0 (*initial*) not yet been busy within the time group, q_1 (*busy*), q_2 (*done*) was busy but is no longer busy, and f is the fail state. The transitions T of the automata are defined as $\{(q_0, 0) \rightarrow q_0, (q_0, 1) \rightarrow q_1, (q_1, 0) \rightarrow q_2, (q_1, 1) \rightarrow q_1, (q_2, 0) \rightarrow$

$q_2, (q_2, 1) \rightarrow f\}$, which enforce that once the resource is busy and then once again idle, it cannot become busy otherwise it ends in a fail state. The final states are $F = \{q_0, q_1, q_2\}$.

$$\texttt{regular}(\big[busy(r,t)|t \in TG\big], S, 0..1, T, s_0, F) \tag{17}$$

The soft constraint version is done by decomposition. The remaining constraints *Avoid Split Assignments*, *Assigned Resources*, *Prefer Resources*, *Limit Workload* are meaningless for the class of problems we examine where all resources are preassigned.

5 Solution-Based Phase Saving

During the search, the CP solver repeatedly makes decisions on which variable and value to branch on. Variables are chosen based on their activity (VSIDS scheme). Phase-saving [23] is almost universally used in SAT solvers, where the choice of value for a decision used is always the value used the last time the variable was seen in search. Solution-based phase saving [1, 4] rather chooses the value for the variable that was used in the last found solution. After a branching variable is selected, the solver is instructed to assign the value that the corresponding variable had in the best solution encountered so far. If that is not possible, it resorts to its default strategy. As a result, the search is focused near the space around the best solution, resembling a large neighbourhood search algorithm, while still remaining complete.

5.1 Hot Starts

At the start of the algorithm, the solver is provided with an initial solution. Due to solution-based phase saving technique, it will immediately focus the search around the given solution. Hence, from the beginning, the search is directed to an area where good solutions reside. As shown in the experimental section, generating the initial solution uses only a small fraction of the total time allocate but our hot start offers significant improvements.

Initial Solution Generation The same procedure for the starting solution is used as in the maxSAT-LNS approach [7]. We briefly outline it.

The main idea is to exploit existing fast heuristic algorithms. To this end, *KHE14*, a publicly available state-of-the-art incomplete algorithm, is first invoked. The method is designed to provide good solutions rapidly. Thus, it is particularly well-suited for our purpose. In the event that KHE14 does not produce a feasible solution, a pure maxSAT approach that treats all split event constraints as hard and ignores soft constraints is called. Afterwards, a simple simulated annealing local search procedure is executed, which attempts to improve the solution by performing two different moves: swaps (exchange the times of two subevents) and block-swap (if a swap move would cause the subevents to overlap, assign to the second one a time such that the two events appear one after the other). During the course of the algorithm only feasible moves between subevents that share at least one resource are considered. The aim is to use inexpensive techniques to remove easy constraint violations, leaving the more challenging ones to the CP solver. We note that initial solution generation takes only a small faction of the total amount of time allocated.

6 Experimental Results

We provide detailed experimentation with the aim of assessing our proposed approach. We have accordingly set the following goals:

– Evaluate the impact of solution-based phase saving and hot starts for high school timetabling. (Section 6.3)
– Test if restarting more frequently would lead to an increase in performance. (Section 6.4)
– Compare the developed approach with other complete methods, namely integer programming and maxSAT. (Section 6.5)
– Position our method among dedicated heuristic algorithms.(Section 6.6)

6.1 Benchmarks and Computing Environment

We considered XHSTT benchmarks from the repository of the International Timetabling Competition 2011 (ITC 2011), limited to those where resources are predefined for events. The majority of the benchmarks fall into this category. Resource assignments would drastically increase the search space if done in a straight-forward manner and further specialized techniques would need to be developed. The maxSAT line of work [7, 8] follows the same restrictions. These datasets include real-world and artificial timetabling problems and an overview is given in Table 2. For more details we refer the interested reader to [24, 25].

We performed all the experiments on an i7-3612QM 2.10 GHz processor with eight GB of RAM, with the exception of the Matheuristic solver (see next section). Each run was given 1200 seconds and a single core, as during the second phase of the competition, with no experiments running in parallel.

6.2 Solvers

Constraint Programming. We used MiniZinc [21] to model XHSTT and Chuffed [3] as the CP solver, for which we implemented solution-based phase saving and the hot start approach. The Luby restart scheme [18] was used within Chuffed.

Complete methods. Integer programming [16] linked with Gurobi 6.5. as the IP solver and the XHSTT-maxSAT formulation [8] with Open-WBO (linear algorithm) [20] as the maxSAT solver.

Dedicated heuristic solvers. KHE14 [15], an ejection-chain-based solver. Variable neighborhood search [11]. Matheuristic [27], an adaptive large neighborhood

Name	$\lvert E\rvert$	$\lvert T\rvert$	$\lvert R\rvert$	$\sum(dur)$
BrazilInstance1	21	25	11	75
BrazilInstance2	63	25	20	150
BrazilInstance3	69	25	24	200
BrazilInstance4	127	25	35	300
BrazilInstance5	119	25	44	325
BrazilInstance6	140	25	44	350
BrazilInstance7	205	25	53	500
FinlandCollege	387	40	111	854
FinlandElementarySchool	291	35	103	445
FinlandHighSchool	172	35	41	297
FinlandSecondarySchool	280	35	64	306
FinlandSecondarySchool2	469	40	79	566
GreeceThirdHighSchoolPatras2010	178	35	113	340
GreeceThirdHighSchoolPreveza2008	164	35	97	340
GreeceWesternUniversityInstance3	210	35	25	210
GreeceWesternUniversityInstance4	262	35	31	262
GreeceWesternUniversityInstance5	184	35	24	184
GreeceFirstHighSchoolAigio2010	283	35	245	532
ItalyInstance1	42	36	16	133
SouthAfricaLewitt2009	185	148	37	838

Table 2. Overview of the datasets used, displaying number of events ($|E|$), times ($|T|$), resources ($|R|$), and sum of event durations ($\sum(dur)$).

search algorithm with integer programming. MaxSAT large neighborhood search algorithm [7] (uses Open-WBO as its maxSAT solver). These results for these methods were averaged over five runs. The Matheuristic solver was not available and therefore the results shown are from the paper [27], which used a benchmarking tool to set fair normalized computation times.

6.3 Phase Saving and Hot Start Impact

We compared three variants of our CP approach: standard (CP), with solution-based phase saving (CP+PS), and with hot starts (CP+HS). Note that hot starts include solution-based phase saving. The results are given in Table 3, showing the soft constraint violations. In addition to the results, we include the value of the hot started solution ("Initial" column in the table).

From the results it is clear that solution-based saving offers improvements over the standard version. The hot start approach further increases the performance. It is of interest to note that in a number of cases the CP solver (with or without phase saving) struggled to find a good solution, but when it was provided with one as a hot start, it managed to provide notable improvements. The additional guidance from phase saving and hot start aids the solver by directing it into fruitful parts of the search space. This approach resembles a local search algorithm, where a better solution is sought for in the vicinity of the best

Name	CP	CP+PS	CP+HS	Initial
Brazil1	52	**41**	**41**	83
Brazil2	102	7	**5**	56
Brazil3	187	43	**25**	116
Brazil4	223	110	**88**	176
Brazil5	355	**32**	57	301
Brazil6	442	74	**66**	192
Brazil7	657	230	**181**	252
FinlandCollege	—	—	**9**	917
FinlandESchool	—	—	**3**	9
FinlandHSchool	314	13	**4**	23
FinlandSSchool	—	—	93	339
FinlandSSchool2	—	—	**1**	7
GreecePatras	—	—	**0**	12
GreecePreveza	—	—	**279**	630
GreeceUni3	—	—	**5**	10
GreeceUni4	—	—	**7**	13
GreeceUni5	—	—	**0**	2
GreeceAigio	—	—	**597**	689
Italy1	—	—	**12**	1229
SAfricaLewitt	—	—	**0**	330

Table 3. Comparison of the three CP variants: CP - standard, CP+PS - with solution-based phase saving, CP+HS - with hot starts, showing the number of soft constraint violations in the best solution found within the time limit. The column "Initial" displays the hot started solution value. Overall, PS+HS is the dominating approach on most benchmarks. No solution generated within the time limit is indicated by '—'.

solution. However, unlike local search, the solution-based phase saving (with hot starts) approach is complete.

6.4 Rapid restarts

Given the similarity between the hot start approach and local search, we decided to experiment with an extreme version of the algorithm by increasing the restart frequency. In our previous experiments, the solver used a Luby restart scheme with the base restart value of 10^4. Now, we set the base restart value to only 100. The results are given in Table 4. The results indicate that rapid restarting does have an impact for a few benchmarks but no conclusive observations can be made. Therefore, it is not considered in further experimentation.

6.5 Comparison of Complete Methods

We compare the hot start CP version with the integer programming and maxSAT approaches in Table 5. The results indicate that the proposed method is indeed effective for high school timetabling. We note that including solution-based phase saving and/or hot starts for the competing methods was not done as it would require modifying the XHSTT solvers to support it.

Name	CP+PS	CP+PS+RR	CP+HS	CP+HS+RR
Brazil1	**41**	**41**	**41**	**41**
Brazil2	7	12	**5**	8
Brazil3	43	26	**25**	26
Brazil4	110	100	88	**85**
Brazil5	32	**30**	57	40
Brazil6	74	103	**66**	77
Brazil7	230	—	**181**	183
FinlandCollege	—	—	**9**	14
FinlandESchool	—	**3**	**3**	**3**
FinlandHSchool	314	13	**4**	23
FinlandSSchool	—	**89**	93	125
FinlandSSchool2	—	—	1	**0**
GreecePatras	—	394	**0**	**0**
GreecePreveza	—	—	279	**140**
GreeceUni3	—	—	**5**	**5**
GreeceUni4	—	—	**7**	**7**
GreeceUni5	—	—	**0**	**0**
GreeceAigio	—	—	bf597	684
Italy1	—	—	**12**	**12**
SAfricaLewitt	—	—	**0**	**0**

Table 4. Analysis of CP variants with rapid restarts (RR in columns). Each entry shows the soft constraint violations. No solution generated is indicated by '—'.

6.6 Comparison with Heuristic Solvers

As previously discussed, solution-based phase saving focuses its search around the current best solution, relating to local search algorithms, while remaining a complete method. Therefore, we decided the evaluate how the approach positions against dedicated heuristic solvers. The results are given in Table 6. Solutions for these randomized (incomplete) methods are averaged over five runs.

Our approach provides competitive results, even when compared against heuristic solvers. Our approach outperforms the competition on most benchmarks, with the exception of the maxLNS approach. When compared to maxLNS, our algorithm outperforms it in two cases. We believe the success of maxLNS is attributed to the fact that it is a dedicated algorithm that incorporates domain-specific knowledge in its search strategy, allowing it to target good solutions quickly. Our approach does not exploit problem-specific details, apart from the initial solution, which both approaches have in common. Nevertheless, it still provides reasonably good results, suggesting that the approach is valuable.

7 Conclusion

We provide a new CP scheduling-based model for the general high school timetabling problem with preassigned resources. We show that significant improvements over the standard CP approach can be obtained by including solution-based phase saving. Further performance increase is achieved by providing the CP solver with a hot start. The resulting approach outperforms other complete approaches and provides competitive results when compared to dedicated heuristic solvers. The techniques used in our approach and maxLNS [7] do not overlap, indicating that a combination of the two approaches might be worth investigating.

Name	IP	maxSAT	CP+HS
Brazil1	41	**39**	41
Brazil2	76	57	**5**
Brazil3	93	75	**25**
Brazil4	234	214	**88**
Brazil5	135	224	**57**
Brazil6	582	352	**66**
Brazil7	1,045	603	**181**
FinlandCollege	1,731	1,309	**9**
FinlandESchool	**3**	**3**	**3**
FinlandHSchool	179	812	**4**
FinlandSSchool	165	504	**93**
FinlandSSchool2	3,505	3,523	**1**
GreecePatras	25	2,329	**0**
GreecePreveza	2,740	5,617	**279**
GreeceUni3	28	7	**5**
GreeceUni4	51	141	**7**
GreeceUni5	3	224	**0**
GreeceAigio	3,738	4,582	**597**
Italy1	15	**12**	**12**
SAfricaLewitt	—	1,039	**0**

Table 5. Comparison of complete methods. No solution generated within the timeout is indicated by '—'.

Name	maxLNS [7]	VNS [11]	KHE14 [15]	CP+HS	Math. [27]
Brazil1	**39**	52.2	54	41	—^{u}
Brazil2	5.4	(1, 44.4)	14	**5**	6
Brazil3	**23**	107.8	116	25	—^{u}
Brazil4	61.4	(17.2, 94.8)	—^{c}	88	**58**
Brazil5	**19.4**	(4, 138.4)	(1, 179)	57	—^{u}
Brazil6	**50.6**	(4, 223.6)	124	66	57
Brazil7	**136.2**	(11.6, 234.6)	179	181	—^{u}
FinlandCollege	54.6	(2.8, 25)	20	**9**	—^{u}
FinlandESchool	**3**	**3**	4	**3**	**3**
FinlandHSchool	9.8	36.6	29	**4**	—^{u}
FinlandSSchool	95.2	(0.4, 93)	**90**	93	—^{u}
FinlandSSchool2	**0.2**	0.2	2	1	6
GreecePatras	**0**	**0**	**0**	**0**	—^{u}
GreecePreveza	38.2	2	**2**	279	—^{u}
GreeceUni3	7	**5**	7	**5**	6
GreeceUni4	**5**	6.2	8	7	12
GreeceUni5	**0**	**0**	**0**	**0**	**0**
GreeceAigio	368	(0.2, 6.2)	**6**	597	180
Italy1	**12**	21.2	31	**12**	—^{u}
SAfricaLewitt	—^{m}	8	—^{c}	**0**	—^{u}

Table 6. Comparison with dedicated heuristic methods. No solution generated within the time limit is indicated by '—'. Solver unavailable listed as '—^{u}'. Insufficient memory noted as '—^{m}'. Values given as pairs (a, b) show the number of hard and soft constraint violations, respectively (for cases where hard constraints are satisfied, only soft constraint violations are displayed). Results averaged over five runs

References

1. Abío Roig, I.: Solving hard industrial combinatorial problems with sat (2013)
2. Global constraint catalog: `all_equal` constraint. http://www.emn.fr/x-info/sdemasse/gccat/Call_equal.html
3. Chu, G.: Improving Combinatorial Optimization. Ph.D. thesis, The University of Melbourne (2011), `http://hdl.handle.net/11343/36679`
4. Chu, G., Stuckey, P.J.: LNS = restarts + dynamic search + phase saving. technical draft
5. Demirović, E., Musliu, N.: Modeling high school timetabling as partial weighted maxSAT. The 4th Workshop on Logic and Search (LaSh 2014)
6. Demirović, E., Musliu, N.: Solving high school timetabling with satisfiability modulo theories. In: Proceedings of the International Conference of the Practice and Theory of Automated Timetabling (PATAT 2014). pp. 142–166
7. Demirović, E., Musliu, N.: MaxSAT based large neighborhood search for high school timetabling. Computers & Operations Research 78, 172–180 (2017)
8. Demirović, E., Musliu, N.: Modeling high school timetabling as partial weighted maxSAT. technical draft - extended LaSh 2014 workshop paper (2017)
9. Global constraint catalog: `disjunctive` constraint. http://www.emn.fr/x-info/sdemasse/gccat/Cdisjunctive.html
10. Dorneles, Á.P., de Araujo, O.C.B., Buriol, L.S.: A fix-and-optimize heuristic for the high school timetabling problem. Computers & Operations Research 52, 29–38 (2014)
11. Fonseca, G.H.G., Santos, H.G.: Variable neighborhood search based algorithms for high school timetabling. Computers & Operations Research 52, 203–208 (2014)
12. da Fonseca, G.H.G., Santos, H.G., Toffolo, T.Â.M., Brito, S.S., Souza, M.J.F.: GOAL solver: a hybrid local search based solver for high school timetabling. Annals of Operations Research 239(1), 77–97 (2016), `http://dx.doi.org/10.1007/s10479-014-1685-4`
13. Jacobsen, F., Bortfeldt, A., Gehring, H.: Timetabling at german secondary schools: tabu search versus constraint programming. In: Proceedings of the International Conference of the Practice and Theory of Automated Timetabling (PATAT 2006)
14. Kheiri, A., Ozcan, E., Parkes, A.J.: HySST: hyper-heuristic search strategies and timetabling. In: Proceedings of the International Conference of the Practice and Theory of Automated Timetabling (PATAT 2012). pp. 497–499
15. Kingston, J.: KHE14: An algorithm for high school timetabling. In: Proceedings of the International Conference of the Practice and Theory of Automated Timetabling (PATAT 2014). pp. 498–501
16. Kristiansen, S., Sørensen, M., Stidsen, T.R.: Integer programming for the generalized high school timetabling problem. Journal of Scheduling 18(4), 377–392 (2015)
17. Lahrichi, A.: Scheduling: the notions of hump, compulsory parts and their use in cumulative problems. C.R. Acad. Sci. Paris 294, 209–211 (1982)
18. Luby, M., Sinclair, A., Zuckerman, D.: Optimal speedup of las vegas algorithms. In: Inf. Proc. Let. vol. 47(4), pp. 173–180 (1993)
19. Marte, M.: Towards constraint-based school timetabling. Annals of Operations Research (ANOR) 155(1), 207–225 (2007)
20. Martins, R., Manquinho, V., Lynce, I.: Open-WBO: a modular maxSAT solver. In: Proceedings of the International Conference on Theory and Applications of Satisfiability Testing (SAT 2014), vol. 8561, pp. 438–445. Springer

21. Nethercote, N., Stuckey, P., Becket, R., Brand, S., Duck, G., Tack, G.: Minizinc: Towards a standard CP modelling language. In: Bessiere, C. (ed.) Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming. LNCS, vol. 4741, pp. 529–543. Springer-Verlag (2007)
22. Pesant, G.: A regular language membership constraint for finite sequences of variables. In: Wallace, M. (ed.) Principles and Practice of Constraint Programming (CP'2004). LNCS, vol. 3258, pp. 482–495. Springer (2004)
23. Pipatsrisawat, K., Darwiche, A.: A lightweight component caching scheme for satisfiability solvers. In: Proceedings of the International Conference on Theory and Applications of Satisfiability Testing (SAT 2007). Lecture Notes in Computer Science, vol. 4501, pp. 294–299. Springer
24. Post, G., Ahmadi, S., Daskalaki, S., Kingston, J., Kyngas, J., Nurmi, C., Ranson, D.: An XML format for benchmarks in high school timetabling. Annals of Operations Research 194(1), 385–397 (2012)
25. Post, G., Kingston, J.H., Ahmadi, S., Daskalaki, S., Gogos, C., Kyngäs, 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)
26. Santos, H.G., Uchoa, E., Ochi, L.S., Maculan, N.: Strong bounds with cut and column generation for class-teacher timetabling. Annals of Operations Research 194(1), 399–412 (2012)
27. Sørensen, M.: A matheuristic for high school timetabling. In: Timetabling at High Schools, PhD thesis. pp. 137–153. Department of Management Engineering, Technical University of Denmark (2013)
28. Sørensen, M., Dahms, F.H.: A two-stage decomposition of high school timetabling applied to cases in Denmark. Computers & Operations Research 43, 36–49 (2014)
29. Sørensen, M., Kristiansen, S., Stidsen, T.R.: International timetabling competition 2011: An adaptive large neighborhood search algorithm. In: Proceedings of the International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012). pp. 489–492
30. Sørensen, M., Stidsen, T.R.: Hybridizing integer programming and metaheuristics for solving high school timetabling. In: Proceedings of the International Conference of the Practice and Theory of Automated Timetabling (PATAT 2014). pp. 557–560
31. Sørensen, M., Stidsen, T.R., Kristiansen, S.: Integer programming for the generalized (high) school timetabling problem. In: Proceedings of the International Conference of the Practice and Theory of Automated Timetabling (PATAT 2014). pp. 498–501 (2014)
32. Sørensen, M., Stidsen, T.R.: High School Timetabling: Modeling and solving a large number of cases in Denmark. In: Proceedings of the International Conference of the Practice and Theory of Automated Timetabling (PATAT 2012). pp. 359–364
33. Sørensen, M., Stidsen, T.R.: Comparing solution approaches for a complete model of high school timetabling. Tech. rep., DTU Management Engineering (2013)
34. Valouxis, C., Housos, E.: Constraint programming approach for school timetabling. Computers and Operations research 30(10), 1555–1572 (2003)