

A Global Constraint for Parallelizing the Execution of Task Sets in Non-Preemptive Scheduling

Michael Marte¹

Institut für Informatik, Universität München
Oettingenstr. 67, 80538 München, Germany
marte@informatik.uni-muenchen.de

1 Introduction

In this paper, we introduce tracks. A track is a global constraint for parallelizing the execution of task sets in non-preemptive scheduling. We give algorithms for bound as well as for domain reasoning, we propose applications in school timetabling, and we report results of a large-scale empirical study. Section 2 introduces concepts and notations we rely on. Section 3 defines tracks in terms of syntax and semantics. Section 4 gives algorithms for solving tracks. Section 5 introduces to school timetabling and section 6 shows how to infer tracks in this setting. Section 7 proposes applications of tracks in school timetabling, presents our experimental design, and reports results. Section 8 closes with perspectives for future work.

2 Preliminaries

Definition 1. A *constraint network* is a triple (X, δ, C) . The single components are specified in the following.

- $X = \{x_1, \dots, x_n\}$ is the set of *problem variables* that constraints may refer to.
- δ is called *computation state*. It is a function from X to sets of values. δ is called *failed* iff $\exists x \in X. \delta(x) = \emptyset$. It is called a *complete assignment* iff $\forall x \in X. |\delta(x)| = 1$. We say that δ_1 *succeeds* δ_0 iff δ_1 is a function on X , $\forall x \in X. \delta_1(x) \subseteq \delta_0(x)$, and $\exists x \in X. \delta_1(x) \subset \delta_0(x)$.
- C is a set of constraints. A *constraint* c is an object that is defined over a set of variables $\text{vars}(c) \subset X$ in terms of the complete assignments that satisfy c .

If $N = (X, \delta, C)$ is a constraint network, then $\text{sol}(N)$ denotes the set of complete assignments that succeed δ and simultaneously satisfy all $c \in C$. Constraint networks $N_0 = (X_0, \delta_0, C_0)$ and $N_1 = (X_1, \delta_1, C_1)$ are called *equivalent* iff $X_0 = X_1$ and $\text{sol}(N_0) = \text{sol}(N_1)$.

¹The author is a PhD student funded by the German Research Council (DFG).

3 Syntax and Semantics of Tracks

Informally, a *track* is of a set of rails, where each *rail* is a set of tasks. The problem of solving a track consists in finding a schedule for each rail such that the schedules cover the same set of time slots.

Definition 2. The expression $\mathbf{track}(T)$ is called a *track* iff $T = \{R_1, \dots, R_n\}$ and

$$\forall 1 \leq i \leq n. \exists m_i > 0. R_i = \{(S_{i1}, P_{i1}), \dots, (S_{im_i}, P_{im_i})\}.$$

(Each pair (S_{ij}, P_{ij}) of problem variables is intended to model a task in terms of its *start time* S_{ij} and its *processing time* P_{ij} . Fixed start or processing times may be modeled by means of variables with singleton domains.)

Definition 3. Let (X, δ, C) be a constraint network such that $\mathbf{track}(T) \in C$ and δ is not failed. Let $R \in T$ and $t = (S, P) \in R$.

1. *Value covers*

(a) $vc(t, \delta) = [\max \delta(S), \min \delta(S) + \min \delta(P) - 1]$

(b) $vc(R, \delta) = \bigcup_{t \in R} vc(t, \delta)$

(c) $vc(T, \delta) = \bigcup_{R \in T} vc(R, \delta)$

2. *Value supplies*

(a) $vs(t, \delta) = \bigcup_{s \in \delta(S)} [s, s + \max \delta(P) - 1]$

(b) $vs(R, \delta) = \bigcup_{t \in R} vs(t, \delta)$

(c) $vs(T, \delta) = \bigcap_{R \in T} vs(R, \delta)$

3. *Earliest start times*

(a) $est(t, \delta) = \min \delta(S)$

(b) $est(R, \delta) = \min_{t \in R} est(t, \delta)$

(c) $est(T, \delta) = \max_{R \in T} est(R, \delta)$

4. *Latest completion times*

(a) $lct(t, \delta) = \max \delta(S) + \max \delta(P) - 1$

(b) $lct(R, \delta) = \max_{t \in R} lct(t, \delta)$

(c) $lct(T, \delta) = \min_{R \in T} lct(R, \delta)$

Definition 4. Let (X, δ, C) be a constraint network such that $\mathbf{track}(T) \in C$. If δ is a complete assignment, $\mathbf{track}(T)$ is satisfied by δ iff

$$|\{vc(R, \delta) : R \in T\}| = 1,$$

i.e. iff the rail schedules cover the same value set.

4 Solving Tracks

We may prune all start and processing times that entail the covering of values that are lower than the earliest start time or greater than the latest completion time of the track (Proposition 1), or that entail the covering of values that are not element of the value supply of the track (Proposition 2). Comparing bounds on the processing times of rails may reveal an inconsistency (Proposition 3). Under certain conditions, a task may be forced to cover a certain value (Proposition 4).

Proposition 1. *Let $N_0 = (X, \delta_0, C)$ be a constraint network such that $\mathbf{track}(T) \in C$ and δ_0 is not failed. Let $R \in T$ and let $(S, P) \in R$. If $\delta_1 = \delta_0$ except for*

$$\begin{aligned} \delta_1(S) &= \{s \in \delta_0(S) : \text{est}(T, \delta_0) \leq s \leq \text{lct}(T, \delta_0) - \min \delta_0(P) + 1\} \\ \text{and } \delta_1(P) &= \{p \in \delta_0(P) : p \leq \text{lct}(T, \delta_0) - \max \{\min \delta_0(S), \text{est}(T, \delta_0)\} + 1\}, \end{aligned}$$

then N_0 and $N_1 = (X, \delta_1, C)$ are equivalent.

Proposition 2. *Let $N_0 = (X, \delta_0, C)$ be a constraint network such that $\mathbf{track}(T) \in C$ and δ_0 is not failed. Let $R \in T$ and let $(S, P) \in R$. If $\delta_1 = \delta_0$ except for*

$$\begin{aligned} \delta_1(S) &= \{s \in \delta_0(S) : \exists p \in \delta_0(P). [s, s + p - 1] \subseteq \text{vs}(T, \delta_0)\} \\ \text{and } \delta_1(P) &= \{p \in \delta_0(P) : \exists s \in \delta_0(S). [s, s + p - 1] \subseteq \text{vs}(T, \delta_0)\}, \end{aligned}$$

then N_0 and $N_1 = (X, \delta_1, C)$ are equivalent.

Proposition 3. *Let $N = (X, \delta, C)$ be a constraint network such that $\mathbf{track}(T) \in C$ and δ is not failed. Let $R_0, R_1 \in T$. Let $l, u \geq 0$ such that, for all $\sigma \in \text{sol}(N)$, l is a lower bound on $|\text{vc}(R_0, \sigma)|$ and u is an upper bound on $|\text{vc}(R_1, \sigma)|$. If $u < l$, then $\text{sol}(N) = \emptyset$.*

Proposition 4. *Let $N_0 = (X, \delta_0, C)$ be a constraint network such that $\mathbf{track}(T) \in C$ and δ_0 is not failed. Let $R \in T$ and let $a \in \text{vc}(T, \delta_0)$ such that $a \notin \text{vc}(R, \delta_0)$. Let $t = (S, P) \in R$ such that $a \in \text{vs}(t, \delta_0)$ and suppose that no other task in R has this property. If $\delta_1 = \delta_0$ except for*

$$\begin{aligned} \delta_1(S) &= \{s \in \delta_0(S) : \exists p \in \delta_0(P). a \in [s, s + p - 1]\} \\ \text{and } \delta_1(P) &= \{p \in \delta_0(P) : \exists s \in \delta_0(S). a \in [s, s + p - 1]\}, \end{aligned}$$

then N_0 and $N_1 = (X, \delta_1, C)$ are equivalent.

5 School Timetabling

The German grammar school (GGS) comprises nine grades (5-13). In grades 5-11, pupils are grouped to form classes. In grades 12 and 13, pupils must choose from a set of courses resulting in a more college-like education. In grades 9-11, several branches of education, differing in curricula, may be available. Frequently, heterogeneous classes with boys and girls from different branches and with different religious denominations cannot be avoided. For economical and educational reasons, heterogeneous classes usually imply the need to join pupils from different classes for physical education, religious

Class	Subjects ^a and Couplings														
7a	K			D	E	L	M	B	G	Ek	Ku	Mu	<i>Sm</i>	<i>Sw</i>	
7b	K	Ev		D	E	L	M	<i>B</i>	<i>G</i>	<i>Ek</i>	<i>Ku</i>	<i>Mu</i>	<i>Sm</i>	<i>Sw</i>	
7c		Ev	Eth	D	E	L	M	<i>B</i>	<i>G</i>	<i>Ek</i>	<i>Ku</i>	<i>Mu</i>	<i>Sm</i>	<i>Sw</i>	
7d	K			D	E	F		<i>M</i>	<i>B</i>	<i>G</i>	<i>Ek</i>	<i>Ku</i>	<i>Mu</i>	<i>Sm</i>	<i>Sw</i>
7e	K	Ev	Eth	D	E	F		<i>M</i>	<i>B</i>	<i>G</i>	<i>Ek</i>	<i>Ku</i>	<i>Mu</i>	<i>Sm</i>	<i>Sw</i>

Table 1: The seventh grade of a timetabling problem

^aK/Ev/Eth = Religious Education for Catholics/Protestants/others, D = German, E = English, F = French, L = Latin, M = Mathematics, B = Biology, G = History, Ek = Geography, Ku = Art, Mu = Music, Sm/Sw = Physical Education for Boys/Girls. If a subject code is printed in italic face, a science lab, a craft room, or some other special facility is required.

Class	Subjects and Tracks													
7a	K			D	E		L	M	B	G	Ek	Ku	Mu	<i>Sm Sw</i>
7b	K	Ev		D	E		L	M	B	G	Ek	Ku	Mu	<i>Sm Sw</i>
7c		Ev	Eth	D	E		L	M	B	G	Ek	Ku	Mu	<i>Sm Sw</i>
7d	K			D	E	F		M	B	G	Ek	Ku	Mu	<i>Sm Sw</i>
7e	K	Ev	Eth	D	E	F		M	B	G	Ek	Ku	Mu	<i>Sm Sw</i>

Table 2: Two tracks inferred from the couplings specified by table 1

education, and branch-specific lessons. The resulting need for simultaneous education of pupils from several classes is modeled by means of couplings. A *coupling* is a set of lessons that are to be scheduled for the same time. Couplings complicate timetabling considerably due to simultaneous resource demands.

Table 1 defines the seventh grade of a GGS timetabling problem in terms of classes, subjects, and couplings. Couplings are marked up by means of bold face types on gray backgrounds of varying intensity where intensity is used to distinguish couplings. We observe that, with respect to gender, all classes are mixed. In fact, this situation results in couplings. For example, the table specifies that all boys from 7b and 7e have to be joined for physical education. The same holds for the girls of 7b and 7e.

In GGS timetabling, the assignment of teachers to lessons is part of the problem specification. In contrast, allocating rooms is part of the timetabling problem. Quality criteria include teacher-specific bounds on the daily workload and subject-specific bounds on the daily number of lessons. Furthermore, tight timeframes are imposed: In grades 5-10, the number of acceptable slots equals the number of lessons prescribed by the curriculum.

6 Inference of Tracks in School Timetabling

In school timetabling, redundant tracks can be inferred from couplings. Before giving our algorithm for inferring tracks, we illustrate the idea by means of two examples.

Table 2 shows two tracks that have been inferred from the couplings that are specified by table 1. Tracks are marked up by means of gray backgrounds of varying intensity where intensity is used to distinguish tracks. For the first example, consider the classes 7b and 7e. Since the pupils are joined for religious and for physical education, the remaining subjects have to be scheduled for the remaining slots. This conclusion has been modeled by a track with two rails. The upper rail contains only lessons of class 7b while the lower rail contains only lessons of class 7e. Each rail contains all lessons of its class except for the lessons pupils from both classes are joined for. Thus, in this case, religious and physical education are missing. For the second example, consider the classes 7a, 7c, and 7d. Since the pupils are joined for physical education, we obtain a track with three rails where physical education is missing.²

Our general inference method works as follows. Let C be a set of classes and suppose that the following conditions hold. (1) The classes in C are joined for m lessons where $m > 0$. (2) The classes in C feature the same timeframe. Let n be the number of slots of this timeframe. (3) For each $c \in C$, the number of lessons prescribed by the curriculum equals n . If these conditions hold, the problem specification implies $\mathbf{track}(T)$ where T contains a rail for each class in C and each rail contains all the lessons of its class except for the lessons pupils from all the classes in C are joined for. Let $R \in T$. We know that $|R| = n - m$. Furthermore, since the lessons of a class must not overlap, $vc(R, \sigma) = |R|$ in any solution σ . In consequence, $|vc(T, \sigma)| = |R|$ in any solution σ .

7 Applications of Tracks in School Timetabling

We exploit tracks in two ways. First, in problem generation to avoid infeasible teacher allocations. Second, in search to prune subtrees without solutions.

Suppose $\mathbf{track}(T)$ has been inferred and let $R \in T$. We know that $|vc(T, \sigma)| = |R|$ in any solution σ . It follows that a teacher must not be assigned more than $|R|$ lessons of T . Otherwise $|vc(T, \sigma)| > |R|$ because the lessons of a teacher must not overlap.

To investigate the operational impact of tracks, we performed an experiment. To ensure the practical relevance of results, the problem set has been generated on the basis of school profiles. A school profile contains key features such as branches, rooms, bounds on the number of pupils, bounds on class sizes, the equipment lessons require, and distributions over the features of pupils and teachers. Table 3 characterizes our schools.

Our basic model features three finite-domain problem variables for each lesson: The domain of the period-level (day-level) slot variable initially contains all the periods (days) of the prescribed timeframe. The domain of the room variable initially contains all the rooms that are acceptable for the lesson. To avoid double-booking of teachers and classes, the model features a period-level, non-preemptive, disjunctive scheduling problem for each teacher and for each class. To enforce quality criteria, the model features a day-level, non-preemptive scheduling problem (either disjunctive or cumulative) for each teacher and for each subject of each class. Furthermore, for each lesson, the model specifies a square that is to be placed in a two-dimensional space spanned by

²At the first glance, this track looks weird because it involves religious education of 7b and 7e, too. This is caused by the coupling in religious education that involves lessons of 7b and 7e.

Feature	School A	School B	School C	School D
Number of Branches	2	5	3	3
Number of Labs, etc.	13	13	24	11
Number of Pupils	350 – 490	610 – 680	940 – 1010	565 – 655
Class Size	13 – 25	16 – 32 ^a	16 – 32	16 – 32
Number of Classes	14 – 21	23 – 25	33 – 36	22 – 23

Table 3: Characteristics of schools

^aIn grade 11: 13 – 25

Problem	Model	School A	School B	School C	School D
Without Tracks		98.2%	43.7%	49.6%	60.1%
With tracks ^a	Without tracks ^b	98.6%	64.1%	68.2%	77.0%
With tracks ^a	With tracks ^c	98.6%	81.1%	65.0%	77.7%

Table 4: Results

^a Tracks have been considered in teacher allocation.

^b Tracks have not been used to prune the search space.

^c Tracks have been used to prune the search space.

the set of periods and by the set of rooms. The position of the square is determined by the period the lesson is scheduled for and by the room that is assigned to the lesson. To avoid double-booking of rooms, the model requires that the squares must not intersect. In addition, symmetries are eliminated by precedences among lessons. For each class and for each subject of that class, a total order among the lessons of the subject is established because, from a practical point of view, the lessons are equivalent.

Our solver embeds constraint propagation into chronological backtracking. Constraint propagation takes place at each node of the search space. The scheduling problems are propagated by means of edge finding [3] and network-flow techniques [4, 5]. The geometric placement problem is propagated by value sweep pruning [1]. Tracks are propagated by means of the domain-reasoning method described by Proposition 2. The search space is unfolded as follows. First, a period-level slot variable is selected. If a dead end was encountered right before, the search procedure will prefer the variable that was selected right before detecting the dead end. Otherwise, one of the variables that have smallest domains is selected. Periods are assigned in ascending order. To maximize the utilization of scarce resources, double lessons are preferably placed following a grid. After the lesson has been placed, a room is assigned. The solver prefers rooms that are as small or supply as few equipment as possible. The solver has been built on top of a finite-domain constraint solver [2] which itself is part of a constraint-logic programming environment.

Table 4 reports the percentage of solved problems (out of 1000) for each class of problems and models. The solver was constrained to abort after 1000 dead ends.

8 Future Work

Future work includes the extension of the problem set, the identification and exploitation of redundant constraints, the optimization of timetables with respect to the number of idle periods by introducing a suitable global constraint, the investigation of discrepancy-based search methods, and the relaxation of unsolvable problems.

References

- [1] N. Beldiceanu. Sweep as a generic pruning technique. Technical Report T2000/08, Swedish Institute of Computer Science, 2000.
- [2] M. Carlsson, G. Ottosson, and B. Carlson. An open-ended finite domain constraint solver. In *Ninth International Symposium on Programming Languages, Implementations, Logics, and Programs*, LNCS 1292, pages 191–206. Springer, 1997.
- [3] P. Martin and D. B. Shmoys. A new approach to computing optimal schedules for the job-shop scheduling problem. In *Proceedings of the 5th International Conference on Integer Programming and Combinatorial Optimization*, LNCS 1084, pages 389–403. Springer, 1996.
- [4] J.-C. Régin. A filtering algorithm for constraints of difference in CSPs. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, volume 1, pages 362–367. AAAI Press, 1994.
- [5] J.-C. Régin. Generalized arc consistency for global cardinality constraint. pages 209–215. AAAI Press/ MIT Press, 1996.