INSTITUT FÜR INFORMATIK
Lehr- und Forschungseinheit für
Programmier- und Modellierungssprachen
Oettingenstraße 67, D–80538 München

Ludwig
Maximilians-
Universität
München

A Global Constraint for Parallelizing the Execution of Task Sets

Michael Marte

http://www.pms.informatik.uni-muenchen.de/publikationen
Forschungsbericht/Research Report PMS-FB-2001-13, Release 30th March 2002

A Global Constraint for Parallelizing the Execution of Task Sets

Michael Marte[*]
Institut für Informatik, Universität München
Oettingenstr. 67, 80538 München, Germany
marte@informatik.uni-muenchen.de

30th March 2002

Abstract

We introduce the *track parallelization problem* (TPP) that consists in parallelizing the execution of task sets. For modelling and solving TPPs in a finite-domain constraint-programming framework, we propose the TPP constraint along with a suitable solver for use in non-preemptive scheduling. We demonstrate how to infer redundant TPPs in school timetabling and we report a large-scale empirical study that has been performed to investigate the effects of TPP propagation in this domain of application.

1 Introduction

Solving a *track parallelization problem* (TPP) consists in parallelizing the execution of task sets. A TPP is specified by a set of tracks where each track is a set of tasks. The problem of solving a TPP then consists in finding a schedule for each track such that the schedules cover the same set of time slots. In its simplest form, a TPP requires to execute two tasks in parallel. In non-preemptive scheduling[1], this can be achieved by equating the start and processing time variables of the tasks. For modelling and solving general TPPs in a finite-domain (FD) constraint-programming (CP) framework, we propose the TPP constraint along with a suitable solver for use in non-preemptive scheduling.

To our best knowledge, the TPP has not yet been described. In terminology, the TPP is related to the *track assignment problem* (TAP) that has been investigated in operations research. A TAP is stated in terms of a task set and a track set. Tasks are specified by intervals, i.e. start and processing times are fixed. A track is either a single machine [4, 16, 10] or a bank of identical parallel machines [8] with time windows of availability. A schedule is an assignment of tasks to tracks that respects the capacity and the availability of the machines. Objectives include the minimization of the number of used tracks [16] and the maximization of the number of scheduled tasks [4, 8]. Applications include the assignment of transmission tasks to satellites [8] and the assignment of clients to summer cottages [10].

The TPP differs from the TAP in several aspects. The chief difference is that solving a TAP requires to assign tracks while solving a TPP requires to assign times such that tracks are executed in parallel. Furthermore, a TAP includes restrictions on the availability of machines

[*]This work was supported by the German Research Council (DFG).
[1]In non-preemptive scheduling, the processing of tasks must not be interrupted.

and the implicit assumption that each task requires a single machine for processing. In a CP fashion, the TPP does not impose any constraint except for that tracks have to be executed in parallel.

TPPs are ubiquitous in school timetabling, especially in settings where options and official regulations imply the need for simultaneous education of pupils from several classes of the same grade. To investigate both the effectiveness of our TPP solver and the operational effects of TPP propagation in school timetabling, we have performed a large-scale empirical study. To ensure the practical relevance of our results, problems close to reality have been generated randomly on the basis of detailed school models. We modeled ten secondary schools and tested 10000 problems. Our solver essentially embeds network-flow techniques and edge finding into chronological backtracking.

This paper is organized as follows. Section 2 introduces some terminology. Section 3 introduces the model of FD constraint solving that our presentation is based on. Section 4 defines the TPP constraint in terms of syntax and semantics. Section 5 proposes a solver for propagating TPP constraints, proves its correctness, and gives performance guarantees like convergence and the ability to decide ground instances. Section 6 introduces to school timetabling, Section 7 shows how to infer TPPs in this setting, and Section 8 presents our empirical study of TPP propagation. Section 9 closes with a summary of what has been achieved.

2 Preliminaries

We start by remembering some concepts that are required to talk about reduction systems and that will be used throughout the paper (cf. [2]).

A *reduction system* is a pair $(A, \rightarrow)$ where A is a set and $\rightarrow \subseteq A \times A$. $\rightarrow^=$ denotes the reflexive closure of $\rightarrow$. $\rightarrow^+$ denotes the transitive closure of $\rightarrow$. $\rightarrow^*$ denotes the reflexive transitive closure of $\rightarrow$. x is called *reducible* iff $\exists y.\, x \rightarrow y$. x is called *in normal form (irreducible)* iff it is not reducible. y is called *a normal form of* x iff $x \rightarrow^* y$ and y is in normal form. We say that y is a *direct successor* of x iff $x \rightarrow y$. We say that y is a *successor* of x iff $x \rightarrow^+ y$. $x, y \in A$ are called *joinable* iff $\exists z.\, x \rightarrow^* z \leftarrow^* y$. We write $x \downarrow y$ to denote that x and y are joinable. $\rightarrow$ is called *terminating* iff there is no chain $a_0 \rightarrow a_1 \rightarrow \ldots$ that descends infinitely. It is called *confluent* iff $y \leftarrow^* x \rightarrow^* z$ implies $y \downarrow z$. It is called *locally confluent* iff $y \leftarrow x \rightarrow z$ implies $y \downarrow z$. It is called *convergent* iff it is terminating and confluent. Let $(A, \rightarrow_1)$ and $(A, \rightarrow_2)$ be reduction systems. We say that $\rightarrow_1$ and $\rightarrow_2$ *commute* iff $y \leftarrow_1^* x \rightarrow_2^* z$ implies $\exists u.\, y \rightarrow_2^* u \leftarrow_1^* z$. We say that $\rightarrow_1$ and $\rightarrow_2$ *commute strongly* iff $y \leftarrow_1 x \rightarrow_2 z$ implies $\exists u.\, y \rightarrow_2^= u \leftarrow_1^* z$.

3 A Model of Finite-Domain Constraint Solving

In this section, we provide a reduction system that captures the process of solving FD constraints. We confined ourselves to solvers where reduction steps transform finite constraint satisfaction problems by pruning values from domains. Neither the addition nor the removal of variables and constraints is supported.

On the conceptual level, a *finite constraint network* (FCN) is a finite (hyper-)graph with variables as nodes and constraints as (hyper-)arcs. Given a FCN, the corresponding *finite constraint satisfaction problem* (FCSP) consists in finding a variable valuation that satisfies all the constraints. We will not distinguish between a FCN and its FCSP.

Let P be a FCSP with variables X and constraints C. We assume that there is a unary constraint for each variable that specifies its set of admissible values. P will be represented by a triple (X,δ,C) where δ is a total function on X (the *domain function* of P) that associates each variable with its set of admissible values (its *domain*). We say that P is *ground* iff all its variables have singleton domains. We say that P is *failed* iff at least one of its variables has an empty domain. We use $\mathrm{scope}(P)$ and $\mathrm{store}(P)$ to denote the variables and constraints of P, respectively. If c is a constraint, $\mathrm{scope}(c)$ denotes the set of variables constrained by c.

Frequently, we will refer to domain functions that have not been declared explicitly. However, in such a case, there will a FCSP the domain function belongs to according to the following naming scheme: If P, P_i, R, R_i, Γ, and Σ denote FCSPs, then δ, δ_i, ρ, ρ_i, γ, and σ are their respective domain functions.

We will consider FCSPs with integer domains only. If a and b are integers, we write $[a,b]$ to denote the set of integers i with $a \leq i \leq b$.

Definition 1. Let $P_0 = (X_0,\delta_0,C_0)$ and $P_1 = (X_1,\delta_1,C_1)$ be FCSPs.

1. $P_0 \rightarrow_{\mathrm{FD}} P_1$ iff $X_1 = X_0$, $C_1 = C_0$, $\delta_1 \neq \delta_0$, and $\delta_1(x) \subseteq \delta_0(x)$ for all $x \in X_0$.
2. $P_1 \in \mathrm{gs}(P_0)$ (P_1 *is a ground successor of* P_0) iff $P_0 \rightarrow_{\mathrm{FD}} P_1$ and P_1 is ground.
3. $P_1 \in \mathrm{sol}(P_0)$ (P_1 *solves* P_0) iff $P_1 \in \mathrm{gs}(P_0)$ and δ_1 simultaneously satisfies all $c \in C$.
4. $P_0 \equiv P_1$ (P_0 *and* P_1 *are equivalent*) iff $\mathrm{sol}(P_0) = \mathrm{sol}(P_1)$.

Corollary 1. $\rightarrow_{\mathrm{FD}}$ *is strict and convergent.*

Corollary 2. *Let* $P_0 = (X,\delta_0,C) \rightarrow_{\mathrm{FD}} P_1$.

1. *Lower (Upper) bounds of domains grow (shrink) monotonically, i.e.* $\min\delta_0(x) \leq \min\delta_1(x) \leq \max\delta_1(x) \leq \max\delta_0(x)$ *for all* $x \in X$.
2. *Sets of ground successors shrink monotonically, i.e.* $\mathrm{gs}(P_0) \supseteq \mathrm{gs}(P_1)$.
3. *Solution sets shrink monotonically, i.e.* $\mathrm{sol}(P_0) \supseteq \mathrm{sol}(P_1)$.

Definition 2. $\rightarrow_r \subseteq \rightarrow_{\mathrm{FD}}$ is called *correct* iff, for all $P_0 \rightarrow_r P_1$, $P_0 \equiv P_1$.

4 Syntax and Semantics of TPP Constraints

A TPP constraint is written as $\mathtt{tpp}(\mathcal{T})$ where $|\mathcal{T}| > 1$ and, for all $T \in \mathcal{T}$, T is a non-empty set of pairs of FD variables. Each pair $(\mathtt{S},\mathtt{P})$ of FD variables is intended to model a task in terms of its *start time* `S` and its *processing time* `P`. Fixed start or processing times may be modeled by means of variables with singleton domains. We assume that processing times are greater than 0.

If P is a FCSP with $\mathtt{tpp}(\mathcal{T}) \in \mathrm{store}(P)$, $T \in \mathcal{T}$, and $t = (\mathtt{S},\mathtt{P}) \in T$, we write $\delta(t)$ instead of $\delta(\mathtt{S}) \times \delta(\mathtt{P})$.

Definition 3. Let P be a FCSP with $\mathtt{tpp}(\mathcal{T}) \in \mathrm{store}(P)$. Let $T \in \mathcal{T}$ and $t = (\mathtt{S},\mathtt{P}) \in T$.

1. *Value covers:*

$$\mathrm{vc}(t,\delta) = \begin{cases} \emptyset, & \text{if } \delta(t) = \emptyset \\ \bigcap\limits_{(s,p)\in\delta(t)} [s, s+p-1] & \text{otherwise} \end{cases}$$

$$\mathrm{vc}(T,\delta) = \bigcup_{t \in T} \mathrm{vc}(t,\delta)$$

$$\mathrm{vc}(\mathcal{T},\delta) = \bigcup_{T \in \mathcal{T}} \mathrm{vc}(T,\delta)$$

2. *Value supplies:*

$$\begin{aligned}
\mathrm{vs}(t,\delta) &= \bigcup_{(s,p)\in\delta(t)} [s, s+p-1] \\
\mathrm{vs}(T,\delta) &= \bigcup_{t\in T} \mathrm{vs}(t,\delta) \\
\mathrm{vs}(\mathcal{T},\delta) &= \bigcap_{T\in\mathcal{T}} \mathrm{vs}(T,\delta)
\end{aligned}$$

Definition 4. Let P be an unfailed FCSP with $\mathtt{tpp}(\mathcal{T}) \in \mathrm{store}(P)$. Let $T \in \mathcal{T}$ and $t = (\mathtt{S},\mathtt{P}) \in T$.

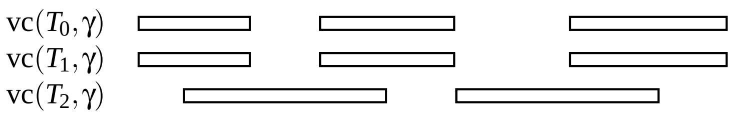
1. *Earliest start times:*

$$\begin{aligned}
\mathrm{est}(t,\delta) &= \min \delta(\mathtt{S}) \\
\mathrm{est}(T,\delta) &= \min_{t\in T} \mathrm{est}(t,\delta) \\
\mathrm{est}(\mathcal{T},\delta) &= \max_{T\in\mathcal{T}} \mathrm{est}(T,\delta)
\end{aligned}$$

2. *Latest completion times:*

$$\begin{aligned}
\mathrm{lct}(t,\delta) &= \max \delta(\mathtt{S}) + \max \delta(\mathtt{P}) - 1 \\
\mathrm{lct}(T,\delta) &= \max_{t\in T} \mathrm{lct}(t,\delta) \\
\mathrm{lct}(\mathcal{T},\delta) &= \min_{T\in\mathcal{T}} \mathrm{lct}(T,\delta)
\end{aligned}$$

The following figure illustrates the concept of parallel execution. We consider the ground problem $(X, \gamma, \{\mathtt{tpp}(\{T_0, T_1, T_2\})\})$. Tracks T_0 and T_1 are executed in parallel because their schedules cover the same value set. In contrast, the schedule of T_3 covers values that the other schedules do not cover; it is not executed in parallel to the other tracks.


$\mathrm{vc}(T_0,\gamma)$
$\mathrm{vc}(T_1,\gamma)$
$\mathrm{vc}(T_2,\gamma)$

Definition 5. Let P be a ground FCSP with $\mathtt{tpp}(\mathcal{T}) \in \mathrm{store}(P)$. δ satisfies $\mathtt{tpp}(\mathcal{T})$ iff

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

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

Value supplies, value covers, earliest start times, and latest completion times have nice monotonicity properties that are summarized in Lemma 1 and Lemma 2. Lemma 3 shows that value supplies are closely related to earliest start and latest completion times. Lemma 4 summarizes properties of ground FCSPs.

Lemma 1. *Suppose* $P_0 \rightarrow_{\mathrm{FD}} P_1$ *and* $\mathtt{tpp}(\mathcal{T}) \in \mathrm{store}(P_0)$. *Let* $T \in \mathcal{T}$ *and* $t \in T$.

1. *Value supplies shrink monotonically, i.e.*
   $\mathrm{vs}(t,\delta_0) \supseteq \mathrm{vs}(t,\delta_1)$, $\mathrm{vs}(T,\delta_0) \supseteq \mathrm{vs}(T,\delta_1)$, *and* $\mathrm{vs}(\mathcal{T},\delta_0) \supseteq \mathrm{vs}(\mathcal{T},\delta_1)$.

2. *Value covers grow monotonically, i.e.*
   $\mathrm{vc}(t,\delta_0) \subseteq \mathrm{vc}(t,\delta_1)$, $\mathrm{vc}(T,\delta_0) \subseteq \mathrm{vc}(T,\delta_1)$, *and* $\mathrm{vc}(\mathcal{T},\delta_0) \subseteq \mathrm{vc}(\mathcal{T},\delta_1)$.

Proof. All properties follow immediately from Corollary 2. □

Lemma 2. *Suppose $P_0 \rightarrow_{\text{FD}} P_1$, P_0 and P_1 are unfailed, and $\texttt{tpp}(\mathcal{T}) \in \text{store}(P_0)$. Let $T \in \mathcal{T}$ and $t \in T$.*

1. *Earliest start times grow monotonically, i.e.*
   $\text{est}(t,\delta_0) \leq \text{est}(t,\delta_1)$, $\text{est}(T,\delta_0) \leq \text{est}(T,\delta_1)$, *and* $\text{est}(\mathcal{T},\delta_0) \leq \text{est}(\mathcal{T},\delta_1)$.

2. *Latest completion times shrink monotonically, i.e.*
   $\text{lct}(t,\delta_0) \geq \text{lct}(t,\delta_1)$, $\text{lct}(T,\delta_0) \geq \text{lct}(T,\delta_1)$, *and* $\text{lct}(\mathcal{T},\delta_0) \geq \text{lct}(\mathcal{T},\delta_1)$.

Proof. All properties follow immediately from Corollary 2. □

Lemma 3. *Suppose P is an unfailed FCSP with $\texttt{tpp}(\mathcal{T}) \in \text{store}(P)$. Let $T \in \mathcal{T}$ and $t = (\texttt{S},\texttt{P}) \in T$.*

1. *Earliest start times are equal to the least elements of value supplies, i.e.*
   $\text{est}(t,\delta) = \min \text{vs}(t,\delta)$, $\text{est}(T,\delta) = \min \text{vs}(T,\delta)$, *and* $\text{est}(\mathcal{T},\delta) = \min \text{vs}(\mathcal{T},\delta)$.

2. *Latest completion times are equal to the greatest elements of value supplies, i.e.*
   $\text{lct}(t,\delta) = \max \text{vs}(t,\delta)$, $\text{lct}(T,\delta) = \max \text{vs}(T,\delta)$, *and* $\text{lct}(\mathcal{T},\delta) = \max \text{vs}(\mathcal{T},\delta)$.

Proof. See Appendix A. □

Lemma 4. *Suppose Γ is a ground FCSP with $\texttt{tpp}(\mathcal{T}) \in \text{store}(\Gamma)$. Let $T \in \mathcal{T}$ and $t \in T$.*

1. *In general,* $\text{vs}(t,\gamma) = \text{vc}(t,\gamma)$ *and* $\text{vs}(T,\gamma) = \text{vc}(T,\gamma)$.

2. *Furthermore, if γ satisfies $\texttt{tpp}(\mathcal{T})$, then*

   (a) $\text{vc}(T,\gamma) = \text{vs}(\mathcal{T},\gamma) = \text{vc}(\mathcal{T},\gamma)$,

   (b) $\text{est}(T,\gamma) = \text{est}(\mathcal{T},\gamma)$, *and*

   (c) $\text{lct}(T,\gamma) = \text{lct}(\mathcal{T},\gamma)$.

Proof. See Appendix A. □

Corollary 3. *Suppose P_0 is a FCSP with $\texttt{tpp}(\mathcal{T}) \in \text{store}(P_0)$. If $P_0 \rightarrow_{\text{FD}} \ldots \rightarrow_{\text{FD}} \Sigma$, Σ is ground, and σ satisfies $\texttt{tpp}(\mathcal{T})$, then the relations depicted in Figure 1 hold.*

5 Solving TPP Constraints

In Section 5.1, we propose four reductions for solving TPP constraints. $\rightarrow_{\text{PVS}}$ identifies and prunes all start and processing times that entail the covering of values that are not element of the value supply of the track set. Under certain conditions, $\rightarrow_{\text{FC}}$ forces tasks to cover values. $\rightarrow_{\text{IPT}}$ reveals inconsistencies by comparing bounds on the processing times of tracks. $\rightarrow_{\text{NC}}$ reveals inconsistencies by identifying situations where values that have to be covered cannot be covered. In Section 5.2, we prove the correctness of this solver. In Section 5.3, we give performance guarantees like convergence and the ability to decide ground instances.

$$
\begin{array}{ccccccccc}
\mathrm{vc}(t,\delta_0) & \subseteq & \dots & \subseteq & \mathrm{vc}(t,\sigma) & = & \mathrm{vs}(t,\sigma) & \subseteq & \dots & \subseteq & \mathrm{vs}(t,\delta_0) \\
\rotatebox{-90}{$\subseteq$} & & & & \rotatebox{-90}{$\subseteq$} & & \rotatebox{-90}{$\subseteq$} & & & & \rotatebox{-90}{$\subseteq$} \\
\mathrm{vc}(T,\delta_0) & \subseteq & \dots & \subseteq & \mathrm{vc}(T,\sigma) & = & \mathrm{vs}(T,\sigma) & \subseteq & \dots & \subseteq & \mathrm{vs}(T,\delta_0) \\
\rotatebox{-90}{$\subseteq$} & & & & \parallel & & \parallel & & & & \rotatebox{-90}{$\subseteq$} \\
\mathrm{vc}(\mathcal{T},\delta_0) & \subseteq & \dots & \subseteq & \mathrm{vc}(\mathcal{T},\sigma) & = & \mathrm{vs}(\mathcal{T},\sigma) & \subseteq & \dots & \subseteq & \mathrm{vs}(\mathcal{T},\delta_0)
\end{array}
$$

and

$$
\begin{array}{ccccccccccc}
\mathrm{est}(t,\delta_0) & \leq & \dots & \leq & \mathrm{est}(t,\sigma) & \leq & \mathrm{lct}(t,\sigma) & \leq & \dots & \leq & \mathrm{lct}(t,\delta_0) \\
\parallel & & & & \parallel & & \parallel & & & & \parallel \\
\min \mathrm{vs}(t,\delta_0) & \leq & \dots & \leq & \min \mathrm{vs}(t,\sigma) & \leq & \max \mathrm{vs}(t,\sigma) & \leq & \dots & \leq & \max \mathrm{vs}(t,\delta_0) \\
\rotatebox{90}{$\leq$} & & & & \rotatebox{90}{$\leq$} & & \rotatebox{-90}{$\leq$} & & & & \rotatebox{-90}{$\leq$} \\
\mathrm{est}(T,\delta_0) & \leq & \dots & \leq & \mathrm{est}(T,\sigma) & \leq & \mathrm{lct}(T,\sigma) & \leq & \dots & \leq & \mathrm{lct}(T,\delta_0) \\
\parallel & & & & \parallel & & \parallel & & & & \parallel \\
\min \mathrm{vs}(T,\delta_0) & \leq & \dots & \leq & \min \mathrm{vs}(T,\sigma) & \leq & \max \mathrm{vs}(T,\sigma) & \leq & \dots & \leq & \max \mathrm{vs}(T,\delta_0) \\
\rotatebox{-90}{$\leq$} & & & & \parallel & & \parallel & & & & \rotatebox{90}{$\leq$} \\
\mathrm{est}(\mathcal{T},\delta_0) & \leq & \dots & \leq & \mathrm{est}(\mathcal{T},\sigma) & \leq & \mathrm{lct}(\mathcal{T},\sigma) & \leq & \dots & \leq & \mathrm{lct}(\mathcal{T},\delta_0) \\
\parallel & & & & \parallel & & \parallel & & & & \parallel \\
\min \mathrm{vs}(\mathcal{T},\delta_0) & \leq & \dots & \leq & \min \mathrm{vs}(\mathcal{T},\sigma) & \leq & \max \mathrm{vs}(\mathcal{T},\sigma) & \leq & \dots & \leq & \max \mathrm{vs}(\mathcal{T},\delta_0)
\end{array}
$$

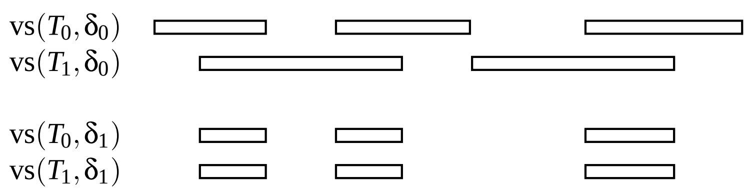
Figure 1: The conclusions of Corollary 3.

5.1 Definition

Definition 6. We say that $P_0 \rightarrow_{\mathrm{PVS}} P_1$ iff $P_0 \rightarrow_{\mathrm{FD}} P_1$ and $\mathtt{tpp}(\mathcal{T}) \in \mathrm{store}(P_0)$, $T \in \mathcal{T}$, $t = (\mathtt{S},\mathtt{P}) \in T$, and $a \in \mathrm{vs}(t,\delta_0)$ exist s.t. $a \notin \mathrm{vs}(\mathcal{T},\delta_0)$ and $\delta_1 = \delta_0$ except for

$$\delta_1(t) = \{(s,p) \in \delta_0(t) : a \notin [s, s+p-1]\}\,.$$

The following figure illustrates the effects of applying $\rightarrow_{\mathrm{PVS}}$ in terms of value supplies. We consider the problem $P_0 = (X, \delta_0, \{\mathtt{tpp}(\{T_0, T_1\})\})$ and its normal form $P_1 = P_0 \downarrow_{\mathrm{PVS}}$.


$\mathrm{vs}(T_0,\delta_0)$
$\mathrm{vs}(T_1,\delta_0)$
$\mathrm{vs}(T_0,\delta_1)$
$\mathrm{vs}(T_1,\delta_1)$

Definition 7. We say that $P_0 \rightarrow_{\mathrm{PVSB}} P_1$ iff $P_0 \rightarrow_{\mathrm{FD}} P_1$, P_0 is not failed, and $\mathtt{tpp}(\mathcal{T}) \in \mathrm{store}(P_0)$, $T \in \mathcal{T}$ and $t \in T$ exist s.t. $\delta_1 = \delta_0$ except for

$$\delta_1(t) = \{(s,p) \in \delta_0(t) : (s,p) \text{ is feasible}\}$$

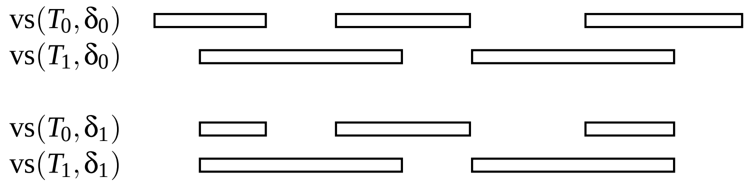
where we say that (s,p) is *feasible* iff

$$\mathrm{est}(\mathcal{T},\delta_0) \le s \le \mathrm{lct}(\mathcal{T},\delta_0) - \min \delta_0(\mathtt{P}) + 1$$

and

$$p \le \mathrm{lct}(\mathcal{T},\delta_0) - \max\{\min \delta_0(\mathtt{S}), \mathrm{est}(\mathcal{T},\delta_0)\} + 1.$$

The following figure illustrates the effects of applying $\rightarrow_{\mathrm{PVSB}}$ in terms of value supplies. We consider the problem $P_0 = (X, \delta_0, \{\mathtt{tpp}(\{T_0, T_1\})\})$ and its normal form $P_1 = P_0 \downarrow_{\mathrm{PVSB}}$.



Definition 8. We say that $P_0 \rightarrow_{\mathrm{FC}} P_1$ iff $P_0 \rightarrow_{\mathrm{FD}} P_1$ and $\mathtt{tpp}(\mathcal{T}) \in \mathrm{store}(P_0)$, $T \in \mathcal{T}$, $t = (\mathtt{S},\mathtt{P}) \in T$, and $a \in \mathrm{vc}(\mathcal{T},\delta_0)$ exist s.t. $a \notin \mathrm{vc}(T,\delta_0)$, $a \in \mathrm{vs}(t,\delta_0)$, $a \notin \mathrm{vs}(u,\delta_0)$ for all $u \in T$, $u \neq t$, and $\delta_1 = \delta_0$ except for

$$\delta_1(t) = \{(s,p) \in \delta_0(t) : a \in [s, s+p-1]\}.$$

Definition 9. We say that $P_0 \rightarrow_{\mathrm{IPT}} P_1$ iff $P_0 \rightarrow_{\mathrm{FD}} P_1$, P_1 is failed, and $\mathtt{tpp}(\mathcal{T}) \in \mathrm{store}(P_0)$, $T_0, T_1 \in \mathcal{T}$, and $l, u \ge 0$ exist s.t., for all $\Gamma \in \mathrm{gs}(P_0)$, l is a lower bound on $|\mathrm{vc}(T_0,\gamma)|$, u is an upper bound on $|\mathrm{vc}(T_1,\gamma)|$, and $u < l$.

For example, consider the problem $(X, \delta, \{\mathtt{tpp}(\{T_0,T_1\})\})$ with $T_0 = \{t_{00}, t_{01}\}$ and $T_1 = \{t_{10}, t_{11}\}$ where $t_{00} = (\{0,5\},\{2\})$, $t_{01} = (\{2,6\},\{1,2,3\})$, $t_{10} = (\{2,3\},\{4,5\})$, and $t_{11} = (\{0,6\},\{2,3\})$ (For simplicity, we replaced the variables by their domains.) We note that $\mathrm{vs}(T_0,\delta) = \mathrm{vs}(T_1,\delta) = [0,8]$ and that T_0 cannot cover more than five values. Since T_0 and T_1 are supplied the same values, $\rightarrow_{\mathrm{PVS}}$ does not apply. If the tasks of T_1 are allowed to overlap, T_1 has a schedule covering five values and $\rightarrow_{\mathrm{IPT}}$ does not apply either. However, if a disjunctive schedule[2] is required for T_1, any schedule of T_1 covers at least six values. In consequence, the tracks cannot be executed in parallel. $\rightarrow_{\mathrm{IPT}}$ will reveal this inconsistency if the demand for disjunctiveness is considered when computing the minimum number of values any schedule of T_1 covers.

Definition 10. Let $P = (X, \delta, C)$ be a FCSP with $\mathtt{tpp}(\mathcal{T}) \in C$. If $T = \{t_1, \ldots t_n\} \in \mathcal{T}$, then $\mathrm{vcg}(\mathcal{T}, T, \delta)$ denotes the bipartite graph (U, V, E) with

- $U = \bigcup\limits_{\substack{1 \le i \le n \\ \delta(\mathtt{P}_i) \neq \emptyset}} \left\{ u_i^j : 0 \le j < \max \delta(\mathtt{P}_i) \right\}$,
- $V = \mathrm{vc}(\mathcal{T}, \delta)$, and
- $E = \left\{ (u_i^j, a) : u_i^j \in U \wedge a \in V \wedge \exists s \in \delta(\mathtt{S}_i).\ s + j = a \right\}$.

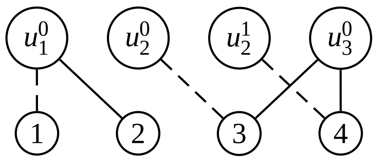
We call this structure *value-cover graph*.

Definition 11. We say that $P_0 \rightarrow_{\mathrm{NC}} P_1$ iff $P_0 \rightarrow_{\mathrm{FD}} P_1$, P_1 is failed, and $\mathtt{tpp}(\mathcal{T}) \in \mathrm{store}(P_0)$ and $T \in \mathcal{T}$ exist s.t. $\mathrm{vcg}(\mathcal{T}, T, \delta_0) = (U, V, E)$ does not have a matching[3] M with $|M| = |V|$.

[2] In a disjunctive schedule, tasks do not overlap.

[3] Given a bipartite graph (U,V,E), a *matching* is a subset of edges $M \subseteq E$ s.t., for all vertices $v \in U \cup V$, at most one edge of M is incident on v.

For example, consider the problem $(X, \delta, \{\mathtt{tpp}(\{T_0, T_1\})\})$ with $\mathrm{vc}(T_0, \delta) = \mathrm{vs}(T_0, \delta) = [1,4]$ and $T_1 = \{t_1, t_2, t_3\}$ where $t_1 = (\{1,2\}, \{1\})$, $t_2 = (\{3\}, \{2\})$, and $t_3 = (\{3,4\}, \{1\})$. (For simplicity, we replaced the variables by their domains.) We note that $\mathrm{vs}(T_1, \delta) = [1,4]$. Hence $\rightarrow_{\mathrm{PVS}}$ does not apply. $\rightarrow_{\mathrm{IPT}}$ does not apply either if starting times are not considered. Now consider the value-cover graph $\mathrm{vcg}(\{T_0, T_1\}, T_1, \delta)$:



The dotted edges constitute a matching of cardinality 3 and it is easy to verify that it has maximum cardinality. Hence only three values out of $[1,4]$ can be covered simultaneously. $\rightarrow_{\mathrm{NC}}$ detects this inconsistency and signals a failure.

5.2 Correctness

Proposition 1. $\rightarrow_{\mathrm{PVS}}$ *is correct.*

Proof. Let $P_0 = (X, \delta_0, C)$ be a FCSP. Let $P_1 = (X, \delta_1, C)$ s.t. $P_0 \rightarrow_{\mathrm{PVS}} P_1$. We have to show that $\mathrm{sol}(P_0) = \mathrm{sol}(P_1)$. By Lemma 2, $\mathrm{sol}(P_1) \subseteq \mathrm{sol}(P_0)$ because $P_0 \rightarrow_{\mathrm{FD}} P_1$. To show that $\mathrm{sol}(P_0) \subseteq \mathrm{sol}(P_1)$, let $\mathtt{tpp}(\mathcal{T}) \in C$, $T \in \mathcal{T}$, $t = (\mathtt{S}, \mathtt{P}) \in T$, and $a \in \mathrm{vs}(t, \delta_0)$ s.t. $a \notin \mathrm{vs}(\mathcal{T}, \delta_0)$ and $\delta_1 = \delta_0$ except for

$$\delta_1(t) = \{(s,p) \in \delta_0(t) : a \notin [s, s+p-1]\}.$$

Obviously,

$$\mathrm{sol}(P_1) = \{\Sigma \in \mathrm{sol}(P_0) : (\mathtt{S}\sigma, \mathtt{P}\sigma) \in \delta_1(t)\}.$$

Let $\Sigma \in \mathrm{sol}(P_0)$ and $(s,p) = (\mathtt{S}\sigma, \mathtt{P}\sigma)$. Suppose $\Sigma \notin \mathrm{sol}(P_1)$, or equivalently, $(s,p) \notin \delta_1(t)$. Hence $a \in [s, s+p-1] = \mathrm{vs}(t, \sigma)$. By Lemma 1, $\mathrm{vs}(\mathcal{T}, \delta_0) \supseteq \mathrm{vs}(\mathcal{T}, \sigma)$. By Lemma 4, $\mathrm{vs}(\mathcal{T}, \sigma) = \mathrm{vs}(T, \sigma)$. Putting it all together we obtain the contradiction

$$a \notin \mathrm{vs}(\mathcal{T}, \delta_0) \supseteq \mathrm{vs}(\mathcal{T}, \sigma) = \mathrm{vs}(T, \sigma) = \bigcup_{t \in T} \mathrm{vs}(t, \sigma) \supseteq \mathrm{vs}(t, \sigma) = [s, s+p-1] \ni a.$$

$\square$

Proposition 2. $\rightarrow_{\mathrm{PVSB}}$ *is correct.*

Proof. Let $P_0 = (X, \delta_0, C)$ be a FCSP. Let $P_1 = (X, \delta_1, C)$ s.t. $P_0 \rightarrow_{\mathrm{PVSB}} P_1$. We have to show that $\mathrm{sol}(P_0) = \mathrm{sol}(P_1)$. By Lemma 2, $\mathrm{sol}(P_1) \subseteq \mathrm{sol}(P_0)$ because $P_0 \rightarrow_{\mathrm{FD}} P_1$. To show that $\mathrm{sol}(P_0) \subseteq \mathrm{sol}(P_1)$, let $\mathtt{tpp}(\mathcal{T}) \in C$, $T \in \mathcal{T}$, and $t \in T$ s.t. $\delta_1 = \delta_0$ except for

$$\delta_1(t) = \{(s,p) \in \delta_0(t) : (s,p) \text{ is feasible}\}$$

Obviously,

$$\mathrm{sol}(P_1) = \{\Sigma \in \mathrm{sol}(P_0) : (\mathtt{S}\sigma, \mathtt{P}\sigma) \text{ is feasible}\}.$$

Let $\Sigma \in \mathrm{sol}(P_0)$ and $(s,p) = (\mathtt{S}\sigma, \mathtt{P}\sigma)$. Suppose $\Sigma \notin \mathrm{sol}(P_1)$, or equivalently, (s,p) is not feasible. Before examining the single cases, it is useful to picture some facts.

- $s = \mathtt{S}\sigma = \min \sigma(\mathtt{S})$ and $p = \mathtt{P}\sigma = \min \sigma(\mathtt{P})$ because σ is a solution.
- By Corollary 2, $\min \sigma(\mathtt{S}) \geq \min \delta_0(\mathtt{S})$ and $\min \sigma(\mathtt{P}) \geq \min \delta_0(\mathtt{P})$ because $\delta_0 \rightarrow_{\mathrm{FD}} \sigma$.

- By Lemma 2, $\mathrm{est}(\mathcal{T},\delta_0) \leq \mathrm{est}(\mathcal{T},\sigma)$ and $\mathrm{lct}(\mathcal{T},\delta_0) \geq \mathrm{lct}(\mathcal{T},\sigma)$ because $\delta_0 \rightarrow_{\mathrm{FD}} \sigma$.
- By Lemma 4, $\mathrm{est}(\mathcal{T},\sigma) = \mathrm{est}(T,\sigma)$ and $\mathrm{lct}(\mathcal{T},\sigma) = \mathrm{lct}(T,\sigma)$ because σ is a solution.
- By Definition 4, $\mathrm{est}(T,\sigma) = \min_{t \in T} \mathrm{est}(t,\sigma) \leq \mathrm{est}(t,\sigma) = \min \sigma(\mathtt{S})$.

We have to consider three cases.

1. $s < \mathrm{est}(\mathcal{T},\delta_0)$: From the facts we collected, we obtain the contradiction
$$\mathrm{est}(\mathcal{T},\sigma) = \mathrm{est}(T,\sigma) \leq \min \sigma(\mathtt{S}) = s < \mathrm{est}(\mathcal{T},\delta_0) \leq \mathrm{est}(\mathcal{T},\sigma).$$
2. $s > \mathrm{lct}(\mathcal{T},\delta_0) - \min \delta_0(\mathtt{P}) + 1$: We note that
$$\begin{aligned}\mathrm{lct}(t,\sigma) &= s + p - 1 \\ &> \mathrm{lct}(\mathcal{T},\delta_0) - \min \delta_0(\mathtt{P}) + p \\ &= \mathrm{lct}(\mathcal{T},\delta_0) - \min \delta_0(\mathtt{P}) + \min \sigma(\mathtt{P}) \\ &\geq \mathrm{lct}(\mathcal{T},\delta_0)\end{aligned}$$
and hence
$$\mathrm{lct}(\mathcal{T},\sigma) = \mathrm{lct}(T,\sigma) = \max_{t \in T} \mathrm{lct}(t,\sigma) \geq \mathrm{lct}(t,\sigma) > \mathrm{lct}(\mathcal{T},\delta_0) \geq \mathrm{lct}(\mathcal{T},\sigma).$$
3. From the facts we collected, we obtain
$$\begin{aligned}p &> \mathrm{lct}(\mathcal{T},\delta_0) - \max\{\min \delta_0(\mathtt{S}), \mathrm{est}(\mathcal{T},\delta_0)\} + 1 \\ &\geq \mathrm{lct}(\mathcal{T},\sigma) - \max\{\min \sigma(\mathtt{S}), \mathrm{est}(\mathcal{T},\sigma)\} + 1 \\ &= \mathrm{lct}(\mathcal{T},\sigma) - \max\{\min \sigma(\mathtt{S}), \mathrm{est}(T,\sigma)\} + 1 \\ &= \mathrm{lct}(\mathcal{T},\sigma) - \min \sigma(\mathtt{S}) + 1 \\ &= \mathrm{lct}(\mathcal{T},\sigma) - s + 1\end{aligned}$$
It follows that $\mathrm{lct}(t,\sigma) = s + p - 1 > \mathrm{lct}(\mathcal{T},\sigma)$. Using this relationship, we obtain the contradiction
$$\mathrm{lct}(\mathcal{T},\sigma) = \mathrm{lct}(T,\sigma) = \max_{t \in T} \mathrm{lct}(t,\sigma) \geq \mathrm{lct}(t,\sigma) > \mathrm{lct}(\mathcal{T},\sigma).$$

□

Proposition 3. *$\rightarrow_{\mathrm{FC}}$ is correct.*

Proof. Let $P_0 = (X, \delta_0, C)$ be a FCSP. Let $P_1 = (X, \delta_1, C)$ s.t. $P_0 \rightarrow_{\mathrm{FC}} P_1$. We have to show that $\mathrm{sol}(P_0) = \mathrm{sol}(P_1)$. By Lemma 2, $\mathrm{sol}(P_1) \subseteq \mathrm{sol}(P_0)$ because $P_0 \rightarrow_{\mathrm{FD}} P_1$. To show that $\mathrm{sol}(P_0) \subseteq \mathrm{sol}(P_1)$, let $\mathtt{tpp}(\mathcal{T}) \in C$, $T \in \mathcal{T}$, $t = (\mathtt{S},\mathtt{P}) \in T$, and $a \in \mathrm{vc}(\mathcal{T},\delta_0)$ s.t. $a \notin \mathrm{vc}(T,\delta_0)$, $a \in \mathrm{vs}(t,\delta_0)$, $a \notin \mathrm{vs}(u,\delta_0)$ for all $u \in T$, $u \neq t$, and $\delta_1 = \delta_0$ except for
$$\delta_1(t) = \{(s,p) \in \delta_0(t) : a \in [s, s+p-1]\}.$$
Obviously,
$$\mathrm{sol}(P_1) = \{\Sigma \in \mathrm{sol}(P_0) : (\mathtt{S}\sigma, \mathtt{P}\sigma) \in \delta_1(t)\}.$$

Let $\Sigma \in \mathrm{sol}(P_0)$ and $(s,p) = (\mathtt{S}\sigma, \mathtt{P}\sigma)$. Suppose $\Sigma \notin \mathrm{sol}(P_1)$, or equivalently, $(s,p) \notin \delta_1(t)$. Hence $a \notin [s, s+p-1] = \mathrm{vc}(t,\sigma)$. Because t was the only task in R that could possibly cover a, we end up with $a \notin \mathrm{vc}(T,\sigma)$. On the other hand side, we know that $a \in \mathrm{vc}(\mathcal{T},\delta_0)$ and thus, by Lemma 1, $a \in \mathrm{vc}(\mathcal{T},\sigma)$. Furthermore, Lemma 4 yields $\mathrm{vc}(\mathcal{T},\sigma) = \mathrm{vc}(T,\sigma)$. We conclude that $a \in \mathrm{vc}(T,\sigma)$. □

Proposition 4. *$\rightarrow_{\text{IPT}}$ is correct.*

Proof. Let $P_0 = (X, \delta_0, C)$ be a FCSP. Let $P_1 = (X, \delta_1, C)$ s.t. $P_0 \rightarrow_{\text{IPT}} P_1$. We have to show that $\text{sol}(P_0) = \text{sol}(P_1)$. Obviously, $\text{sol}(P_1) = \emptyset$. Let $\Sigma \in \text{sol}(P_0)$. We know that $\texttt{tpp}(\mathcal{T}) \in C$, $T_0, T_1 \in \mathcal{T}$, and $l, u \geq 0$ exist s.t. l is a lower bound on $|\text{vc}(T_0, \sigma)|$, u is an upper bound on $|\text{vc}(T_1, \sigma)|$, and $u < l$. Furthermore, by Lemma 4, we know that $\text{vc}(T_0, \sigma) = \text{vc}(T_1, \sigma) = \text{vc}(\mathcal{T}, \sigma)$. Putting it all together we obtain the contradiction

$$|\text{vc}(\mathcal{T}, \sigma)| = |\text{vc}(T_1, \sigma)| \leq u < l \leq |\text{vc}(T_0, \sigma)| = |\text{vc}(\mathcal{T}, \sigma)|.$$

□

Lemma 5. *Let $P = (X, \delta, C)$ be a FCSP s.t. $\texttt{tpp}(\mathcal{T}) \in C$. Let $T \in \mathcal{T}$. If $\text{sol}(P) \neq \emptyset$, then $\text{vcg}(\mathcal{T}, T, \delta)$ has a matching M with $|M| = |V|$.*

Proof. Let $(U, V, E) = \text{vcg}(\mathcal{T}, T, \delta)$ (cf. Definition 10), $\Sigma \in \text{sol}(P)$, and

$$M = \left\{ (u_i^{a - \texttt{S}_i\sigma}, a) : 1 \leq i \leq n \wedge a \in V \wedge a \in \text{vc}(t_i, \sigma) \wedge \forall 1 \leq j < i.\ a \notin \text{vc}(t_j, \sigma) \right\}.$$

We will show that $M \subseteq E$, M is a matching, and $|M| = |V|$.

1. $M \subseteq E$: Let $(u_i^j, a) \in M$. From the definition of M, we know that $1 \leq i \leq n$, $j = a - \texttt{S}_i\sigma$, $a \in V$, and $a \in \text{vc}(t_i, \sigma)$. First, we have to show that $s \in \delta(\texttt{S}_i)$ exists s.t. $s + j = a$. Let $s = \texttt{S}_i\sigma$. $s + j = \texttt{S}_i\sigma + a - \texttt{S}_i\sigma = a$ and, by Lemma 2, $s \in \delta(\texttt{S}_i)$. It remains to show that $u_i^j \in U$, or equivalently, $0 \leq j < \max \delta(\texttt{P}_i)$. From $a \in \text{vc}(t_i, \sigma) = [\texttt{S}_i\sigma, \texttt{S}_i\sigma + \texttt{P}_i\sigma - 1]$ and by Lemma 2, we conclude that
$$0 \leq a - \texttt{S}_i\sigma = j \leq \texttt{P}_i\sigma - 1 < \texttt{P}_i\sigma = \max \sigma(\texttt{P}_i) \leq \max \delta(\texttt{P}_i).$$
2. M is a matching: Let $(u_i^j, a) \in M$.
   (a) u_i^j is the only mate of a in M: Suppose $(u_k^l, a) \in M$ s.t. $(k, l) \neq (i, j)$. We note that $j = l = a - \texttt{S}_i\sigma$ which implies $k \neq i$. Suppose $k < i$. Then $(u_i^j, a) \notin M$ because $a \in \text{vc}(t_k, \sigma)$. Suppose $k > i$. Then $(u_k^l, a) \notin M$ because $a \in \text{vc}(t_i, \sigma)$.
   (b) a is the only mate of u_i^j in M: Suppose $(u_i^j, b) \in M$ s.t. $b \neq a$. However, by the definition of M, $a = j + \texttt{S}_i\sigma = b$.
3. $|M| = |V|$: Let $a \in V = \text{vc}(\mathcal{T}, \delta)$. By Lemma 1, $a \in \text{vc}(\mathcal{T}, \sigma)$ and, by Lemma 4, $a \in \text{vc}(T, \sigma)$. We have to show that $1 \leq i \leq n$ exists such that $(u_i^{a - \texttt{S}_i\sigma}, a) \in M$. Let $I = \{1 \leq i \leq n : a \in \text{vc}(t_i, \sigma)\}$. $I \neq \emptyset$ because otherwise $a \notin \text{vc}(T, \sigma) = \bigcup_{1 \leq i \leq n} \text{vc}(t_i, \sigma)$. Let $i = \min I$. It is easy to verify that $(u_i^{a - \texttt{S}_i\sigma}, a) \in M$. Furthermore, $u_i^{a - \texttt{S}_i\sigma}$ is the only mate of a in M because M is a matching. Thus $|M| = |V|$.

□

Proposition 5. *$\rightarrow_{\text{NC}}$ is correct.*

Proof. Let $P_0 = (X, \delta_0, C)$ be a FCSP. Let $P_1 = (X, \delta_1, C)$ s.t. $P_0 \rightarrow_{\text{NC}} P_1$. We have to show that $\text{sol}(P_0) = \text{sol}(P_1)$. Obviously, $\text{sol}(P_1) = \emptyset$. Suppose $\text{sol}(P_0) \neq \emptyset$. By Lemma 5, $\text{vcg}(\mathcal{T}, T, \delta_0) = (U, V, E)$ has a matching M with $|M| = |V|$ for all $\texttt{tpp}(\mathcal{T}) \in C$ and $T \in \mathcal{T}$. □

5.3 Performance Guarantees

Proposition 6. *If $R \subseteq \{\rightarrow_{\text{PVS}}, \rightarrow_{\text{FC}}, \rightarrow_{\text{IPT}}, \rightarrow_{\text{NC}}\}$ and $\rightarrow_r = \bigcup R$, then $\rightarrow_r$ is convergent.*

Proof. See [11]. $\square$

Lemma 6. *Let $(A, \rightarrow)$ be a reduction system. If $\rightarrow$ is convergent, then each $a \in A$ has a unique normal form.*

Proof. See [2]. $\square$

Corollary 4. *Let P be a FCSP. If $R \subseteq \{\rightarrow_{\text{PVS}}, \rightarrow_{\text{FC}}, \rightarrow_{\text{IPT}}, \rightarrow_{\text{NC}}\}$ and $\rightarrow_r = \bigcup R$, then P has a unique normal form wrt. $\rightarrow_r$.*

Proposition 7. *Let $P_0 = (X, \delta_0, C)$ be a ground, unfailed FCSP s.t. $\mathtt{tpp}(\mathcal{T}) \in C$. If δ_0 violates $\mathtt{tpp}(\mathcal{T})$, then a failed FCSP P_1 exists s.t. $P_0 \rightarrow_{\text{PVS}} P_1$.*

Proof. We know that $|\{\text{vc}(T, \delta_0) : T \in \mathcal{T}\}| > 1$, because $\mathcal{T} \neq \emptyset$ and δ_0 violates $\mathtt{tpp}(\mathcal{T})$. Let $T_0, T_1 \in \mathcal{T}$ s.t. $\text{vc}(T_0, \delta_0) \neq \text{vc}(T_1, \delta_0)$. If $\text{vc}(T_0, \delta_0) \subset \text{vc}(T_1, \delta_0)$, let $a \in \text{vc}(T_1, \delta_0) - \text{vc}(T_0, \delta_0)$ and $t = (\mathtt{S}, \mathtt{P}) \in T_1$ s.t. $a \in \text{vc}(t, \delta_0)$. Otherwise, let $a \in \text{vc}(T_0, \delta_0) - \text{vc}(T_1, \delta_0)$ and $t = (\mathtt{S}, \mathtt{P}) \in T_0$ s.t. $a \in \text{vc}(t, \delta_0)$. Let $P_1 = (X, \delta_1, C)$ with $\delta_1 = \delta_0$ except for $\delta_1(t) = \emptyset$. Obviously, P_1 is failed and $P_0 \rightarrow_{\text{FD}} P_1$. It remains to verify that $a \in \text{vs}(t, \delta_0)$, $a \notin \text{vs}(\mathcal{T}, \delta_0)$, and

$$\{(s, p) \in \delta_0(t) : a \notin [s, s + p - 1]\} = \emptyset.$$

The latter equation holds because $(\mathtt{S}\delta_0, \mathtt{P}\delta_0)$ is the only candidate and, by assumption, $a \in \text{vc}(t, \delta_0) = [\mathtt{S}\delta_0, \mathtt{S}\delta_0 + \mathtt{P}\delta_0 - 1]$. $a \in \text{vs}(t, \delta_0)$ because, by Lemma 4, $\text{vs}(t, \delta_0) = \text{vc}(t, \delta_0)$ and, by assumption, $a \in \text{vc}(t, \delta_0)$. $a \notin \text{vs}(\mathcal{T}, \delta_0)$ because, by Definition 3 and by Lemma 4,

$$\text{vs}(\mathcal{T}, \delta_0) = \bigcap_{T \in \mathcal{T}} \text{vs}(T, \delta_0) \subseteq \text{vs}(T_0, \delta_0) \cap \text{vs}(T_1, \delta_0) = \text{vc}(T_0, \delta_0) \cap \text{vc}(T_1, \delta_0)$$

and a has been chosen s.t. $a \notin \text{vc}(T_0, \delta_0) \cap \text{vc}(T_1, \delta_0)$. $\square$

Proposition 8. *Let $R \subseteq \{\rightarrow_{\text{PVS}}, \rightarrow_{\text{FC}}, \rightarrow_{\text{IPT}}, \rightarrow_{\text{NC}}\}$ s.t. $\rightarrow_{\text{PVS}} \in R$ and let $\rightarrow_r = \bigcup R$. Let $P_0 = (X, \delta_0, C)$ be a ground, unfailed FCSP s.t. $\mathtt{tpp}(\mathcal{T}) \in C$. If δ_0 violates $\mathtt{tpp}(\mathcal{T})$, then $P_0 \downarrow_r$ is failed.*

Proof. By Proposition 7, a failed FCSP P_1 exists s.t. $P_0 \rightarrow_{\text{PVS}} P_1$. We know that $P_1 \downarrow_r = P_0 \downarrow_r$ and thus $P_0 \rightarrow_{\text{PVS}} P_1 \rightarrow_r^* P_0 \downarrow_r$. It follows that $P_1 \rightarrow_{\text{FD}}^* P_0 \downarrow_r$ because $\rightarrow_r \subseteq \rightarrow_{\text{FD}}$. We conclude that $P_0 \downarrow_r$ is failed. $\square$

Proposition 9. *Let $P_0 = (X, \delta_0, C)$ be a ground, unfailed FCSP s.t. $\mathtt{tpp}(\mathcal{T}) \in C$. If δ_0 violates $\mathtt{tpp}(\mathcal{T})$, then a failed FCSP P_1 exists s.t. $P_0 \rightarrow_{\text{NC}} P_1$.*

Proof. We know that $|\{\text{vc}(T, \delta_0) : T \in \mathcal{T}\}| > 1$, because $\mathcal{T} \neq \emptyset$ and δ_0 violates $\mathtt{tpp}(\mathcal{T})$. Let $T_0, T_1 \in \mathcal{T}$ s.t. $\text{vc}(T_0, \delta_0) \neq \text{vc}(T_1, \delta_0)$. Let $G = (U, V, E) = \text{vcg}(\mathcal{T}, T_0, \delta_0)$ and let M be a matching in G. Considering the structure of $\text{vcg}(\mathcal{T}, T_0, \delta_0)$ (cf. Definition 10), we find that

$$\{a : \exists u \in U.\ (u, a) \in E\} = \text{vc}(T_0, \delta_0).$$

Thus, $|\text{vc}(T_0, \delta_0)|$ is an upper on $|M|$. We obtain

$$|M| \leq |\text{vc}(T_0, \delta_0)| < |\text{vc}(T_0, \delta_0) \cup \text{vc}(T_1, \delta_0)| \leq \left| \bigcup_{T \in \mathcal{T}} \text{vc}(T, \delta_0) \right| = |\text{vc}(\mathcal{T}, \delta_0)| = |V|$$

and conclude that G does not have a matching M with $|M| = |V|$. Thus, for any failed FCSP P_1 s.t. $P_0 \rightarrow_{\text{FD}} P_1$, $P_0 \rightarrow_{\text{NC}} P_1$. $\square$

Proposition 10. *Let $R \subseteq \{\rightarrow_{\text{PVS}}, \rightarrow_{\text{FC}}, \rightarrow_{\text{IPT}}, \rightarrow_{\text{NC}}\}$ s.t. $\rightarrow_{\text{NC}} \in R$ and let $\rightarrow_r = \bigcup R$. Let $P_0 = (X, \delta_0, C)$ be a ground, unfailed FCSP s.t. $\texttt{tpp}(\mathcal{T}) \in C$. If δ_0 violates $\texttt{tpp}(\mathcal{T})$, then $P_0 \downarrow_r$ is failed.*

Proof. Similar to the proof of Proposition 8 by employing Proposition 9. □

6 School Timetabling

The German *Gymnasium* is a secondary school that provides nine grades (5–13) of academically orientated education. In grades 5–11, pupils are grouped to form classes. In grades 12 and 13, pupils must choose from a set of course combinations resulting in a more college-like education. In grades 9–11, several branches of education, differing in curricula, may be available. 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 official regulations. Drexl & Salewski stated that “the situation we are confronted with in secondary schools in Germany (...) seems to be the most general one which has ever been addressed in the open literature.” [7].

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 split classes and/or to join pupils from different classes for physical education, religious education, and branch-specific lessons. The resulting need for parallel education of several groups complicates timetabling considerably due to simultaneous resource demands.

In the following, we show how to model requests for splitting classes and/or joining pupils from different classes by means of modules.

Definition 12. $((P_i)_{1 \leq i \leq g}, s, (d_j)_{1 \leq j \leq n})$ is called a *module* iff $g > 0$, the P_i are non-empty, disjoint pupil sets, s is a subject code, $n > 0$, and the d_j are positive integers.

Suppose $\mathcal{M} = ((P_i)_{1 \leq i \leq g}, s, (d_j)_{1 \leq j \leq n})$ is a module. For each $1 \leq i \leq g$, $\mathcal{M}$ requires to give n lessons $(l_{ij})_{1 \leq j \leq n}$ of s to P_i a week where lesson l_{ij} lasts d_j periods; $(d_j)_{1 \leq j \leq n}$ encodes the mode $d_1 - \ldots - d_n$. Thus $\mathcal{M}$ gives rise to gn lessons that occupy $g\Sigma_{1 \leq j \leq n} d_j$ slots.

Note that a module does not require parallel education. If this is required, the module has to be wrapped into a coupling.

Definition 13. Suppose $c = (\mathcal{M}_i)_{1 \leq i \leq m}$ is a family of modules. c is called a *coupling* iff $m > 0$ and, for all $1 \leq i, j \leq m$, $\mathcal{M}_i$ and $\mathcal{M}_j$ have the same mode.

A coupling requires to parallelize the education of the sections of its modules.

In practice, it is the duty of the headmaster to specify modules and couplings. When doing this, the headmaster has to keep to official regulations [1] that for each grade, branch, and subject specify the subject matter and the number of periods to teach a week. In particular, those regulations restrict the possibilities for joint education of pupils from different branches.

Tables 1 and 2 define the seventh grade and the tenth grade, respectively, of a timetabling problem. If there is no requirement to teach the subject s to the class c, then the field (c, s) is empty. If the subject s has to be taught to the class c as a whole, then the field (c, s) specifies the weekly number of periods of education. Otherwise, if it is necessary to split classes and/ or to join pupils from different classes, the corresponding fields contain at least a small letter that

Class[b]	REC[c]	REP	Eth[d]	G	E	F[e]	L	M	**B**	H	Geo	**A**	**Mu**	**PEM**[f]	**PEF**	#[g]
7a-EL[h]	2			4	4		5	4	2	2	1	2	2	$2b^2_3$	b	30
7b-EL	$2a^3_5$	a		4	4		5	4	2	2	1	2	2	$2c^2_3$	c	30
7c-EL		a	a	4	4		5	4	2	2	1	2	2	b	b	30
7d-EF	a			4	4	5		4	2	2	1	2	2	$2d^2_2$	d	30
7e-EF	a	a	a	4	4	5		4	2	2	1	2	2	c	c	30

Table 1:
The seventh grade of a timetabling problem.

[a]Subject Codes: REC/REP = Religious Education for Catholics/Protestants, Eth = Ethics, G = German, E = English, F = French, L = Latin, M = Mathematics, B = Biology, H = History, Geo = Geography, A = Art, Mu = Music, PEM/PEF = Physical Education for Boys/Girls. If a subject code is printed in bold face, a science lab, a craft room, or some other special facility is required.

[b]There are two options in foreign-language education: EF and EL. EF (EL) denotes that English is taught as first foreign language – starting from the fifth grade – and French (Latin) is taught as second foreign language – starting from the seventh grade. We observe that all classes are pure with respect to the second foreign language.

[c]By default, all Catholic pupils attend REC and all Protestant pupils attend REP.

[d]All pupils that are neither Catholic nor Protestant attend Eth. Furthermore, parents of a Catholic or Protestant child may require that their child attends Eth instead of REC or REP.

[e]No couplings arise from options in foreign-language education because all classes are pure to this respect.

[f]In this school, genders are segregated in physical education.

[g]For each class, this column gives the total number of periods per week.

[h]7a is a pure Catholic class while all other classes are mixed with respect to religious denomination.

Class[b]	RE[c]	G	E	F	L	M	**Ph**	**C**	**B**	(H & Soc)[d]	EL	**A**[e]	**Mu**	**PE**[c]	**HE**	#
10a/ML-ELF	$2a^3_4$	$3b^1_1$	$3e^1_1$	5	$3i^2_2$	$3j^1_2$	$2l^1_2$		$2p^1_1$	$3s^1_1$	$1v^1_1$	$1y^2_4$	y	$2z^2_2$		30
10a/SOC-EL	a	b	e		i	j	l	$2n^1_1$	p	$4t^1_1$	v	y	y	z	$2\beta^1_1$	30
10a/SOC-EF	a	b	e	i		j	l	n	p	t	v	y	y	z	β	30
10b/SOC-EF	a	$3c^1_1$	$3f^1_1$	$3h^1_1$		j	l	n	$2q^1_1$	t	$1w^1_1$	y	y	$2\alpha^2_2$	β	30
10b/NAT-EF	a	c	f	h		$4k^1_1$	$3m^1_1$	$3o^1_1$	q	s	w	y	y	α		30
10c/NAT-EF	a	$3d^1_1$	$3g^1_1$	i		k	m	o	$2r^1_1$	$3u^1_1$	$1x^1_1$	y	y	z		30
10c/NAT-EL	a	d	g		i	k	m	o	r	u	x	y	y	z		30

Table 2:
The tenth grade of a timetabling problem.

[a]In addition to the subject codes introduced in Table 1, we use the following subject codes: RE = Religious Education, Ph = Physics, C = Chemistry, Soc = Social Studies, EL = Economics and Law, PE = Physical Education, HE = Home Economics

[b]Before the ninth grade, pupils have to opt for a branch of education. In this case, Modern Languages (ML), Social Sciences (SOC), and Natural Sciences (NAT) are offered. ML may be chosen only if Latin was chosen as second foreign language. In ML, French is taught as third foreign language. Note that the pupils of 10a/ML-ELF, 10a/SOC-EL, and 10a/SOC-EF make up the class 10a. However, in this case, they only attend nine periods a week together without some pupils leaving or joining the class.

[c] For reasons of space and clarity, we merged REC, REP, and Eth into RE, and PEM and PEF into PE.

[d]History is taught two periods a week in the first term and one period a week in the second term. Depending on the branch, Social Studies is taught one or two periods a week in the first term and two or three periods a week in the second term.

[e]Pupils must opt for either Art or Music.

uniquely identifies the coupling that is used to model the resulting request for parallel education of several groups. Furthermore, for each coupling, there is a field (in our presentation, the left-and/or top-most one) that contains ni_g^m where i uniquely identifies the coupling, m specifies the number of its modules, g specifies the number of groups to teach in parallel, and n specifies the weekly number of periods of education. For example, $2c_3^2$ contained in the field (7b-EL, PEM) of Table 1 denotes that three groups from two modules have to be educated in parallel and that each group has to taught for two periods a week. Actually, this coupling has been imposed to facilitate the segregation of genders in physical education while minimizing personnel costs. Note that the implementation of c requires to find two time slots such that both classes involved, three teachers, and three sports facilities are available.

In practice, couplings emanate from different classes having the same, tight timeframe. For example, reconsider the coupling c from Table 1. Suppose that coupling c has two boy groups and one girl group. Consider a boy group that has a pupil from 7b-EL. If this boy group and the girl group are not educated in parallel, then at least 31 slots are required to schedule the lessons of 7b-EL. Since this would violate the tight timeframe of 30 slots, both groups have to be educated in parallel. The same reasoning applies to the second boy group.

7 Inference of TPPs in School Timetabling

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

Figure 2 shows two TPPs that have been inferred from the couplings that are specified by Table 1. TPPs are marked up by means of gray backgrounds of varying intensity where intensity is used to distinguish tracks. The first TPP stems from the classes 7b-EL and 7e-EF. Since the pupils are joined for religious and for physical education and timeframes are tight, the remaining subjects have to be scheduled for the remaining slots. This conclusion has been modeled by a TPP with two tracks. The upper track contains all lessons that do involve 7b-EL but not 7e-EF while the lower track contains all lessons that do involve 7e-EF but not 7b-EL. Each track contains all lessons its class is involved in except for the lessons both classes are involved in. Thus, in this case, religious and physical education are missing. The second TPP stems from the classes 7a-EL and 7c-EL. Since the pupils are joined for physical education only, we obtain a TPP with two tracks where physical education is missing. At the first glance, the track of class 7c-EL looks weird because it involves religious education of 7b-EL, 7d-EF, and 7e-EF, too. This is consequence of coupling a which requires to parallelize religious education in 7b-EL, 7c-EL, 7d-EF, and 7e-EF.

Figure 3 shows two TPPs that have been inferred from the couplings that are specified by Table 2. The first TPP stems from the classes 10a/ML-ELF and 10a/SOC-EL. It essentially reflects the curricular differences of the branches Modern Languages (ML) and Social Sciences (SOC). While ML pupils are taught a third foreign language and Social Studies, SOC pupils are taught Chemistry, Social Studies, and Home Economics. The second TPP stems from the classes 10b/SOC-EF and 10b/NAT-EF. As the previous TPP, it reflects the curricular differences of two branches. Pupils in Natural Sciences (NAT) attend ten lessons of Mathematics, Physics, and Chemistry a week. In the SOC branch, only seven lessons a week are allocated for these subjects. In return, SOC pupils attend more lessons in Social Studies than their NAT colleagues and receive education in Home Economics. The third TPP stems from the classes 10a/SOC-EF and 10b/SOC-EF. This TPP is basically due to pupils from the same branch but from different

Class	REC	REP	Eth	G	E	F	L	M	**B**	H	Geo	**A**	**Mu**	**PEM**	**PEF**	#[a]
7a-EL																
7b-EL				4	4		5	4	2	2	1	2	2			26
7c-EL																
7d-EF																
7e-EF				4	4	5		4	2	2	1	2	2			26
7a-EL	2			4	4		5	4	2	2	1	2	2			28
7b-EL	$2a_5$	a														
7c-EL		a	a	4	4		5	4	2	2	1	2	2			28
7d-EF	a															
7e-EF	a	a	a													

Figure 2:
Two TPPs inferred from the couplings specified by Table 1. The upper TPP stems from the classes 7b-EL and 7e-EF. The lower TPP stems from the classes 7a-EL and 7c-EL.

[a]For each track, this column gives the number of periods per week.

Class	RE	G	E	F	L	M	**Ph**	**C**	**B**	(H & Soc)	EL	**A**	**Mu**	**PE**	**HE**	#[a]
10a/ML-ELF				5						$3s_1$						8
10a/SOC-EL								$2n_1$		$4t_1$					$2\beta_1$	8
10a/SOC-EF								n		t					β	
10b/SOC-EF								n		t					β	
10b/NAT-EF										s						
10c/NAT-EF																
10c/NAT-EL																
10a/ML-ELF						$3j_2$	$2l_2$			$3s_1$						
10a/SOC-EL						j	l	$2n_1$		$4t_1$					$2\beta_1$	
10a/SOC-EF						j	l	n		t					β	
10b/SOC-EF						j	l	n		t					β	13
10b/NAT-EF						$4k_1$	$3m_1$	$3o_1$		s						13
10c/NAT-EF						k	m	o								
10c/NAT-EL						k	m	o								
10a/ML-ELF		$3b_1$	$3e_1$		$3i_2$				$2p_1$		$1v_1$			$2z_2$		
10a/SOC-EL		b	e		i				p		v			z		
10a/SOC-EF		b	e	i					p		v			z		14
10b/SOC-EF		$3c_1$	$3f_1$	$3h_1$					$2q_1$		$1w_1$			$2\alpha_2$		14
10b/NAT-EF		c	f	h					q		w			α		
10c/NAT-EF				i										z		
10c/NAT-EL					i									z		

Figure 3:
Three TPPs inferred from the couplings specified by Table 2. The first TPP stems from the the classes 10a/ML-ELF und 10a/SOC-EL. The second TPP stems from the classes 10b/SOC-EF and 10b/NAT-EF. The third TPP stems from the classes 10a/SOC-EF and 10b/SOC-EF.

[a]For each track, this column gives the number of periods per week.

classes being joined for branch-specific education[4].

Our algorithm for TPP inference is based on the following observation. Let C be a set of classes and suppose 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 official regulations equals n.

If these conditions hold, the problem specification implies the TPP $\mathcal{T}$ where $\mathcal{T}$ contains a track for each class in C and each track contains all the lessons its class is involved in except for the lessons all the classes in C are involved in.

This observation immediately suggests a straight-forward method for TPP inference: For each grade and for each subset of its classes, try to generate a TPP as described. Although this approach has exponential complexity, it is feasible in practice.

8 An Empirical Study of TPP Propagation In School Timetabling

To investigate the operational effects of TPP propagation in school timetabling, we have performed a large-scale empirical study. More precisely, we investigated how TPP propagation affects the probability that a problem can be solved and which combination of propagation rules performs best. Due to limited computational resources, we did not employ a full factorial design but confined ourselves to the separate investigation of $\rightarrow_{\text{PVS}}$, $\rightarrow_{\text{PVSB}}$, $\rightarrow_{\text{FC}}$, $\rightarrow_{\text{NC}}$, and $\bigcup\{\rightarrow_{\text{PVS}}, \rightarrow_{\text{FC}}, \rightarrow_{\text{NC}}\}$. We did not investigate $\rightarrow_{\text{IPT}}$ because, in our application, the processing times of tracks are fixed.

To ensure the practical relevance of our results, problems close to reality have been generated[5] randomly on the basis of detailed models of representative schools. A school model describes the options offered to the pupils, the facilities, the teacher population, the pupil population, and various requirements of the school management that concern the design of timetables. We modeled ten secondary schools (R1–R6, A1–A4) that are briefly described in Table 3. Schools R1–R6 have real-world counterparts while schools A1–A4 are artificial schools that were created by hybridising schools int the following way.

- A1 hybridises R2 and R3. The facilities and the pupil numbers are inherited from R2, the study options are inherited from R3.

- A2 hybridises R2 and R3. The facilities and the pupil numbers are inherited from R3, the study options are inherited from R2.

- A3 hybridises R2 and R5. The facilities and the pupil numbers are inherited from R2. A3 study options all the options its parents offer.

- A4 hybridises R3 and A3. The facilities and the pupil numbers are inherited from R3, the study options are inherited from A3.

For each school, we generated and tested 1000 problems. In problem generation, we tried to avoid infeasible teacher allocations by considering couplings.

[4]Note that, in general, it is not the case that a lesson is member of at most one track. Furthermore, TPPs with more than two tracks are frequent.

[5]The problem generator will be described in detail in a future report.

Feature	R1	R2	R3	R4	R5	R6	A1	A2	A3	A4
#Branches[a]	2	5	3	3	4	7	3	5	7	7
#Labs[b]	17	19	23	13	13	18	19	23	19	23
#Classrooms	23	25	45	26	38	47	25	45	25	45
#Pupils	450–630	675–761	1120–1210	685–795	715–805	985–1075	675–761	1120–1210	675–761	1120–1210
Group Size	13–25	16–32[c]	16–32	16–32[c]	16–32	16–32[c]	16–32	16–32	16–32	16–32
#Classes[d]	14–21	22–23	33–36	22–23	23–25	29–30	22–23	33–36	22–23	33–36
#Pupils*[e]	540	718	1165	740	760	1030	718	1165	718	1165
#Teachers*	57	64	91	63	64	86	63	94	64	93
#Modules*	319	328	447	328	316	401	329	451	330	454
#Lessons*	707	779	1157	790	798	1054	785	1157	786	1151
#Couplings*	263	278	387	269	274	338	277	395	281	400
#TPPs*	11	20	30	22	17	40	15	34	27	36
Teachers' Workload[f]*	17	17	17	17	17	17	17	17	17	17
#Tasks*	591	631	900	600	625	764	636	911	646	933
#Variables*	925	994	1422	963	985	1239	999	1441	1009	1454
#Scheduling Problems*	477	487	669	479	479	620	483	680	498	687
#Involved Resources*	5	5	5	6	5	6	5	5	6	5

Table 3: Characteristics of schools, problems, and models (from top to bottom).

[a]'#Branches' is short for 'The Number of Branches'.
[b]Including craft rooms, sports facilities, etc.
[c] In grade 11: 12–24
[d]The number of classes in grades 5–11.
[e]If a row is annotated with *, then, in this row, distributions are described in terms of their medians.
[f]A teacher's workload is measured in periods per week.

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 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[6] among lessons. For each class and for each subject of that class, a total order among the lessons of the subject can be established because, from a practical point of view, the lessons are equivalent. TPPs are mapped to global TPP constraints except for couplings which are enforced in the following way. Suppose $c = \|(\mathcal{M}_i)_{1\leq i\leq m}\|$ is a coupling such that all its modules have the same mode $(d_k)_{1\leq k\leq n}$. Let $1 \leq i \leq m$ and suppose $\mathcal{M}_i$ has groups $(P_j)_{1\leq j\leq g}$. For each of its groups, $\mathcal{M}_i$ gives rise to n lessons. Let $1 \leq j \leq g$ and let $(l_{ijk})_{1\leq k\leq n}$ denote the lessons of P_j. For each $1 \leq k \leq n$, l_{11k} is used instead of l_{ijk} throughout the model. l_{11k} exists because all modules have the same mode. In consequence, μ_1 has to make sure that all resources required for processing l_{ijk} are available when l_{11k} is processed. This is accomplished by adjusting the resource requirements of l_{11k}.

Our timetabling engine embeds constraint propagation into chronological backtracking. At each node of the search space, a task is chosen and scheduled and rooms are allocated. All decisions required to unfold the search space are guided by strategies. Constraint propagation takes place after each commitment that is issued to the constraint solver and is performed in a fixed-point manner. The scheduling problems are propagated by means of edge finding[7] [12] and network-flow techniques[8] [14, 15]. The geometric placement problem is propagated by value sweep pruning [3]. Our TPP solver implements $\rightarrow_{\text{PVS}}$, $\rightarrow_{\text{PVSB}}$, $\rightarrow_{\text{FC}}$, and $\rightarrow_{\text{NC}}$. The implementation of $\rightarrow_{\text{NC}}$ is based on the Ford and Fulkerson algorithm (e.g. [13]). The propagation rules can be applied in any combination and the solver can be switched off completely. In task selection, we exploit heuristic knowledge gained from prior failures. If there are unscheduled tasks that were met at a dead end before (in a chronological sense, not with respect to the path from the root of the search tree to the current node), only this task set is subject to selection. Otherwise, all unscheduled tasks are subject to selection. The search procedure prefers tasks the period-level slot variables of which have smallest domains. Ties are broken by considering the processing time and the number of resources required; tasks that maximize the product of processing time and resource demand are prefered. Periods are assigned in ascending order.

[6] A *precedence* is a constraint over two tasks that states that either task must precede the other one.

[7] *Edge finders* are algorithms that, given a search state and a task set, derive precedences among the tasks that each schedule reachable from the search state has to satisfy. We employed the edge finder built into the `serialized` and `cumulative` constraints as provided by SICStus Prolog 3.9.0 [9]. This edge finder is based on [12].

[8] More precisely, we used the `all_distinct` and `global_cardinality` constraints as provided by SICStus Prolog 3.9.0 [9] that maintain domain consistency for alldiff and global cardinality constraints, respectively. An *alldiff constraint* [18] specifies that all its variables must take pairwise distinct values. [14] proposes an efficient algorithm to maintain domain consistency for alldiff constraints. For each value, a global cardinality constraint (gcc) imposes bounds on the number of variables that the value may be assigned to. The gcc generalizes the alldiff constraint. [15] presents an efficient algorithm to maintain domain consistency[9] for gccs.

[9] Suppose $P = (X, \delta, C)$ is a FCSP and $c = p(x_1, \ldots, x_n) \in C$. According to van Hentenryck et al. [17], c is *domain-consistent*, if, for each variable x_i and for each value $v_i \in \delta(x_i)$, there exist values $v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n$ in $\delta(x_1), \ldots, \delta(x_{i-1}), \delta(x_{i+1}), \ldots, \delta(x_n)$ s.t. $p(v_1, \ldots, v_n)$ holds.

	S1	S2	S3	S4	S5	S6
R1	77.5 (2.64)	77.5 (2.64)	77.5 (2.64)	77.5 (2.64)	77.5 (2.64)	77.5 (2.64)
R2	71.8 (2.85)	83.9 (2.33)	75.0 (2.74)	71.9 (2.84)	84.1 (2.31)	75.0 (2.74)
R3	82.0 (2.43)	83.3 (2.36)	83.4 (2.35)	82.0 (2.43)	83.3 (2.36)	83.4 (2.35)
R4	78.2 (2.61)	79.5 (2.55)	78.6 (2.60)	78.2 (2.61)	79.5 (2.55)	78.6 (2.60)
R5	87.4 (2.10)	86.8 (2.14)	87.6 (2.09)	87.4 (2.10)	86.8 (2.14)	87.6 (2.09)
R6	62.0 (3.07)	60.6 (3.09)	61.3 (3.08)	62.0 (3.07)	60.6 (3.09)	61.3 (3.08)
A1	85.8 (2.21)	86.7 (2.15)	86.1 (2.19)	85.8 (2.21)	86.7 (2.15)	86.1 (2.19)
A2	74.6 (2.75)	84.0 (2.32)	76.6 (2.68)	74.6 (2.75)	83.9 (2.33)	76.6 (2.68)
A3	66.2 (2.99)	71.7 (2.85)	65.8 (3.00)	66.2 (2.99)	71.6 (2.85)	65.8 (3.00)
A4	63.7 (3.04)	72.4 (2.83)	64.2 (3.03)	63.7 (3.04)	72.4 (2.83)	64.2 (3.03)

Table 4: Results ($100\bar{x}$ with $200\hat{\sigma}_{\bar{x}}\%$ in parentheses)

If the task has even duration (double lessons), odd periods are prefered to maximize the utilization of scarce resources. In room allocation, the search procedure prefers rooms that are as small or supply as few equipment as possible. The timetabling engine has been built on top of a finite-domain constraint solver [5] which itself is part of the constraint-logic programming environment SICStus Prolog 3.9.0 [9].

We tested six solvers that differ in how TPP constraints are propagated. S1 does not propagate TPP constraints while, according to our experimental design outlined earlier, S2–S6 employ $\rightarrow_{\mathrm{PVS}}$, $\rightarrow_{\mathrm{PVSB}}$, $\rightarrow_{\mathrm{FC}}$, $\rightarrow_{\mathrm{NC}}$, and $\bigcup\{\rightarrow_{\mathrm{PVS}}, \rightarrow_{\mathrm{FC}}, \rightarrow_{\mathrm{NC}}\}$, respectively.

Table 4 reports the percentage of solved problems out of 1000 together with the 95% confidence intervals[10]. Runs were constrained to abort after 1000 dead ends. Due to the large sample size, the 95% confidence intervals are quite narrow. We make the following observations:

- TPP propagation pays off for the schools R2, A2, A3, and A4.
- $\rightarrow_{\mathrm{PVS}}$ is effective for each of these schools while $\rightarrow_{\mathrm{PVSB}}$ is effective for school R2 only.
- The other reductions do not contribute, not even when applied in combination.
- In no case, the robustness[11] of our timetabling engine was considerably impaired by TPP propagation.

Appendix B has plots that give more insight into the search process by relating the number of solved problems

- to the number of backtracking steps that were necessary to obtain a solution;
- to the number of deadends that were encountered during search.

[10] Suppose we are given a sample with mean $\bar{x}$ and standard error $\hat{\sigma}_{\bar{x}}$. Then the *95% confidence interval* (e.g. [6]) is approximately $\bar{x} \pm 2\hat{\sigma}_{\bar{x}}$. This interval is likely to contain the population mean μ with 95% probability. For example, according to Table 4, S1 solved $\bar{x} = 77.5\%$ of the R1 problems. In this case, $2\hat{\sigma}_{\bar{x}} \approx 2.64\%$. This means that we can be 95% sure that, in the long term, S1 will be able to solve $77.5\% \pm 2.64\%$ of the R1 problems.

[11] The term *robustness* refers to the probability of solving a problem from a certain problem class. For example, we say that solver S_0 is more robust than solver S_1 wrt. to a certain problem class C, if S_0 is more likely to solve problems from C than S_1.

9 Summary

We introduced the track parallelization problem (TPP) and we presented a global finite-domain constraint for track parallelization along with a suitable solver. Furthermore, we demonstrated how to infer redundant TPPs in school timetabling and we reported a large-scale empirical study that has been performed to investigate the effects of TPP propagation in this domain of application. We found that, for certain problem classes, the robustness of our timetabling engine is considerably increased by propagating redundant TPPs.

A Proofs

Proof to Lemma 3.

1. We observe that

$$\begin{aligned}
\operatorname{est}(t,\delta) &= \min \delta(\mathtt{S}) = \min_{s \in \delta(\mathtt{S})} s \\
&= \min_{s \in \delta(\mathtt{S})} \min\, [s, s + \max \delta(\mathtt{P}) - 1] \\
&= \min \bigcup_{s \in \delta(\mathtt{S})} [s, s + \max \delta(\mathtt{P}) - 1] \\
&= \min \bigcup_{(s,p) \in \delta(t)} [s, s + p - 1] = \min \operatorname{vs}(t,\delta).
\end{aligned}$$

   This result yields

$$\begin{aligned}
\operatorname{est}(T,\delta) &= \min_{t \in T} \operatorname{est}(t,\delta) = \min_{t \in T} \min \operatorname{vs}(t,\delta) = \min \bigcup_{t \in T} \operatorname{vs}(t,\delta) \\
&= \min \operatorname{vs}(T,\delta).
\end{aligned}$$

   It follows that

$$\begin{aligned}
\operatorname{est}(\mathcal{T},\delta) &= \max_{T \in \mathcal{T}} \operatorname{est}(T,\delta) = \max_{T \in \mathcal{T}} \min \operatorname{vs}(T,\delta) = \min \bigcap_{T \in \mathcal{T}} \operatorname{vs}(T,\delta) \\
&= \min \operatorname{vs}(\mathcal{T},\delta).
\end{aligned}$$

2. We observe that

$$\begin{aligned}
\operatorname{lct}(t,\delta) &= \max \delta(\mathtt{S}) + \max \delta(\mathtt{P}) - 1 \\
&= \max_{s \in \delta(\mathtt{S})} (s + \max \delta(\mathtt{P}) - 1) \\
&= \max_{s \in \delta(\mathtt{S})} \max\, [s, s + \max \delta(\mathtt{P}) - 1] \\
&= \max \bigcup_{s \in \delta(\mathtt{S})} [s, s + \max \delta(\mathtt{P}) - 1] \\
&= \max \bigcup_{(s,p) \in \delta(t)} [s, s + p - 1] = \max \operatorname{vs}(t,\delta).
\end{aligned}$$

   This result yields

$$\begin{aligned}
\operatorname{lct}(T,\delta) &= \max_{t \in T} \operatorname{lct}(t,\delta) = \max_{t \in T} \max \operatorname{vs}(t,\delta) = \max \bigcup_{t \in T} \operatorname{vs}(t,\delta) \\
&= \max \operatorname{vs}(T,\delta).
\end{aligned}$$

   It follows that

$$\begin{aligned}
\operatorname{lct}(\mathcal{T},\delta) &= \min_{T \in \mathcal{T}} \operatorname{lct}(T,\delta) = \min_{T \in \mathcal{T}} \max \operatorname{vs}(T,\delta) = \max \bigcap_{T \in \mathcal{T}} \operatorname{vs}(T,\delta) \\
&= \max \operatorname{vs}(\mathcal{T},\delta).
\end{aligned}$$

□

Proof to Lemma 4.

1. We observe that

$$\begin{aligned}\mathrm{vs}(t,\gamma) &= \bigcup_{(s,p)\in\gamma(t)} [s, s+p-1] \\ &= [\mathtt{S}\gamma, \mathtt{S}\gamma + \mathtt{P}\gamma - 1] \\ &= \bigcap_{(s,p)\in\gamma(t)} [s, s+p-1] = \mathrm{vc}(t,\gamma).\end{aligned}$$

   It follows that

$$\mathrm{vs}(T,\gamma) = \bigcup_{t\in T} \mathrm{vs}(t,\gamma) = \bigcup_{t\in T} \mathrm{vc}(t,\gamma) = \mathrm{vc}(T,\gamma).$$

2. Suppose γ satisfies $\mathtt{tpp}(\mathcal{T})$.

   (a) Definition 5 implies that $\mathrm{vc}(\mathcal{T},\gamma) = \mathrm{vc}(T,\gamma)$. It follows that

$$\mathrm{vs}(\mathcal{T},\gamma) = \bigcap_{T\in\mathcal{T}} \mathrm{vs}(T,\gamma) = \bigcap_{T\in\mathcal{T}} \mathrm{vc}(T,\gamma) = \bigcap_{T\in\mathcal{T}} \mathrm{vc}(\mathcal{T},\gamma) = \mathrm{vc}(\mathcal{T},\gamma).$$

   (b) From Lemma 3 we know that $\mathrm{est}(T,\gamma) = \min \mathrm{vs}(T,\gamma)$. It follows that

$$\begin{aligned}\mathrm{est}(\mathcal{T},\gamma) &= \max_{T\in\mathcal{T}} \mathrm{est}(T,\gamma) = \max_{T\in\mathcal{T}} \min \mathrm{vs}(T,\gamma) = \max_{T\in\mathcal{T}} \min \mathrm{vs}(\mathcal{T},\gamma) \\ &= \min \mathrm{vs}(\mathcal{T},\gamma) = \min \mathrm{vs}(T,\gamma) = \mathrm{est}(T,\gamma).\end{aligned}$$

   (c) From Lemma 3 we know that $\mathrm{lct}(T,\gamma) = \max \mathrm{vs}(T,\gamma)$. It follows that

$$\begin{aligned}\mathrm{lct}(\mathcal{T},\gamma) &= \min_{T\in\mathcal{T}} \mathrm{lct}(T,\gamma) = \min_{T\in\mathcal{T}} \max \mathrm{vs}(T,\gamma) = \min_{T\in\mathcal{T}} \max \mathrm{vs}(\mathcal{T},\gamma) \\ &= \max \mathrm{vs}(\mathcal{T},\gamma) = \max \mathrm{vs}(T,\gamma) = \mathrm{lct}(T,\gamma).\end{aligned}$$

□

B Behaviour of Solvers

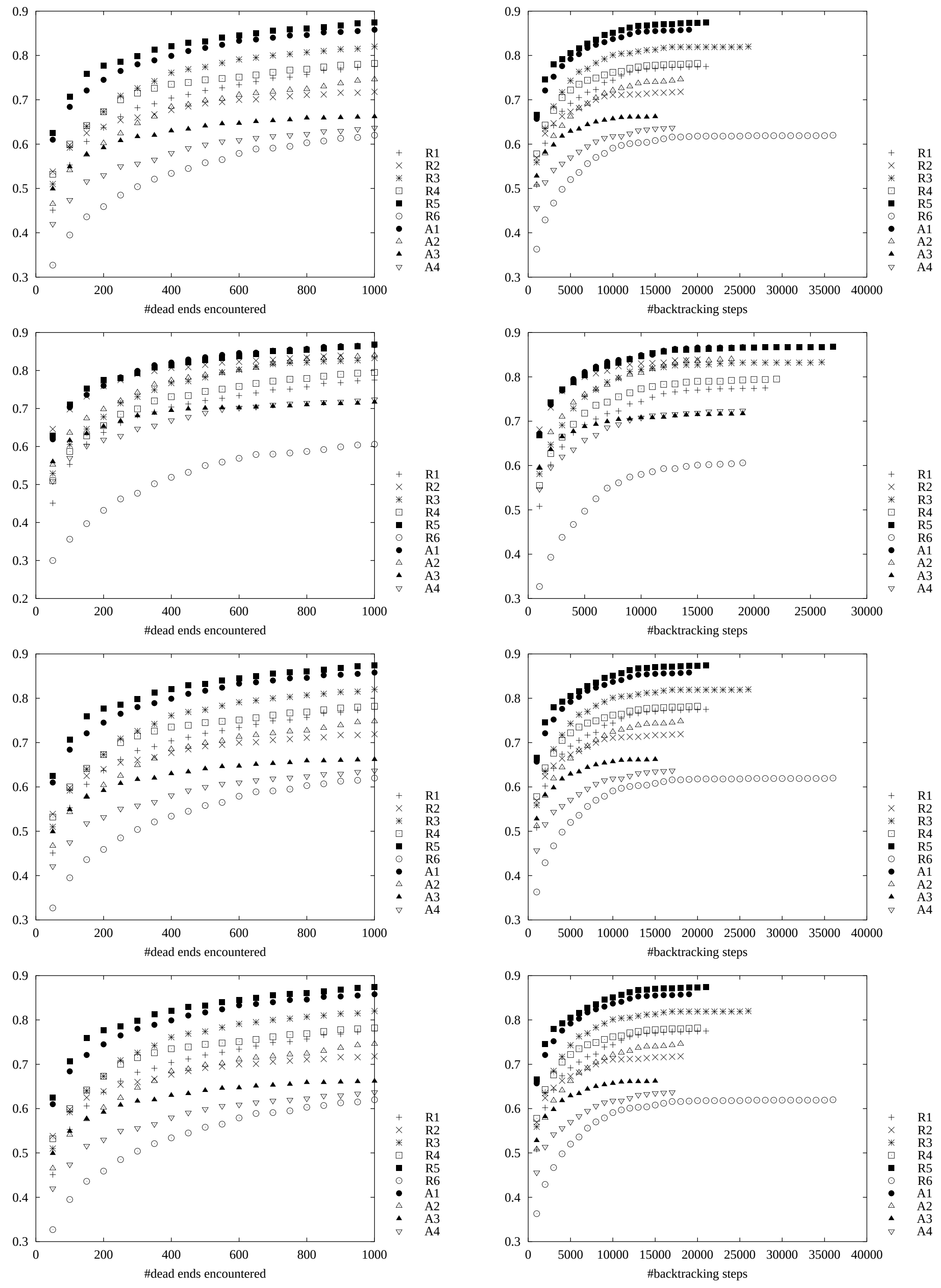


Figure 4: Behaviour of solvers S1–S4 (from top to bottom)

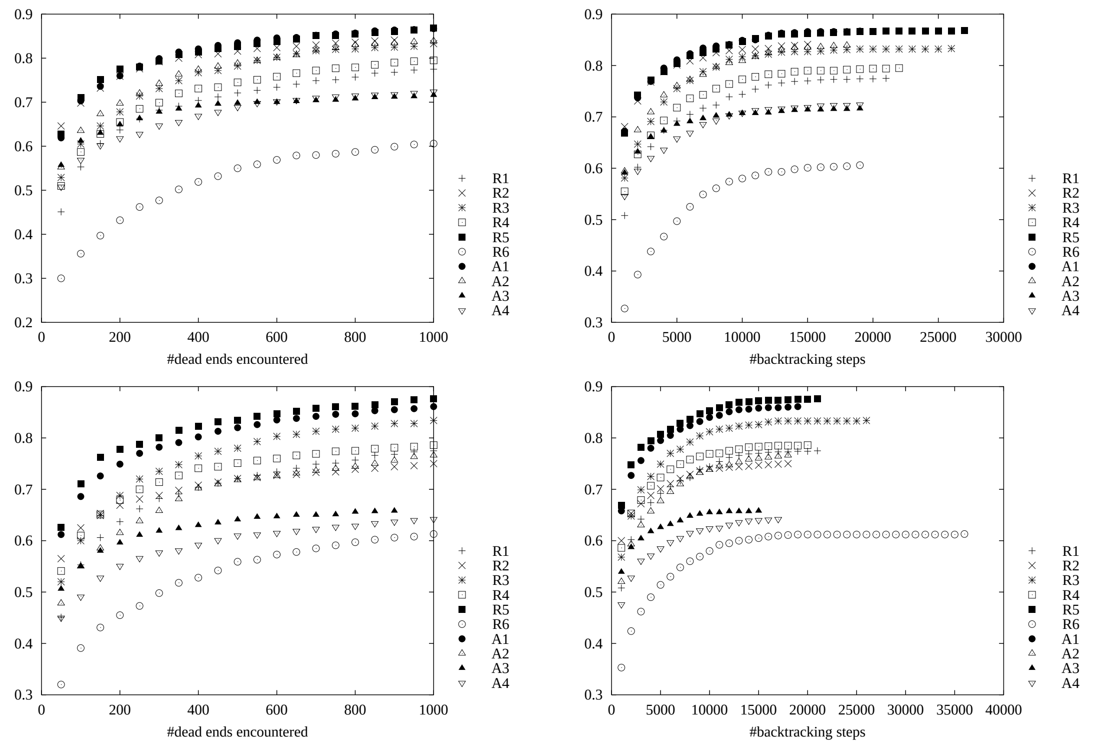


Figure 5: Behaviour of solvers S5 and S6 (from top to bottom)

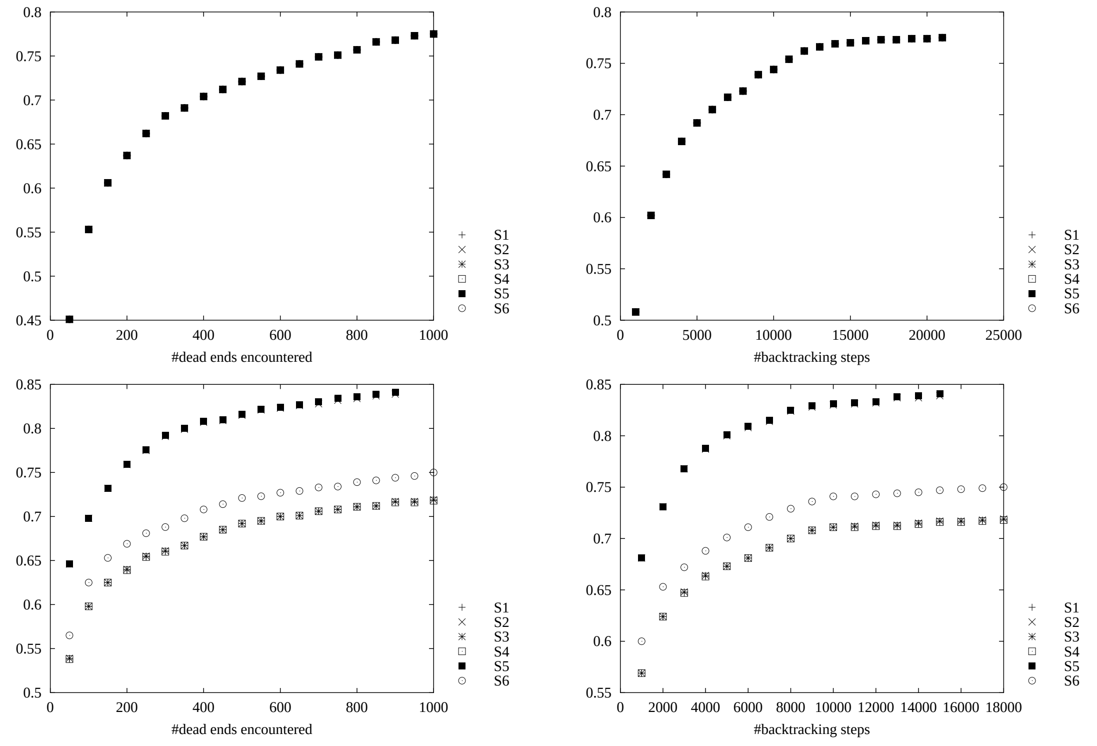


Figure 6: Behaviour of solvers for schools R1 and R2 (from top to bottom)

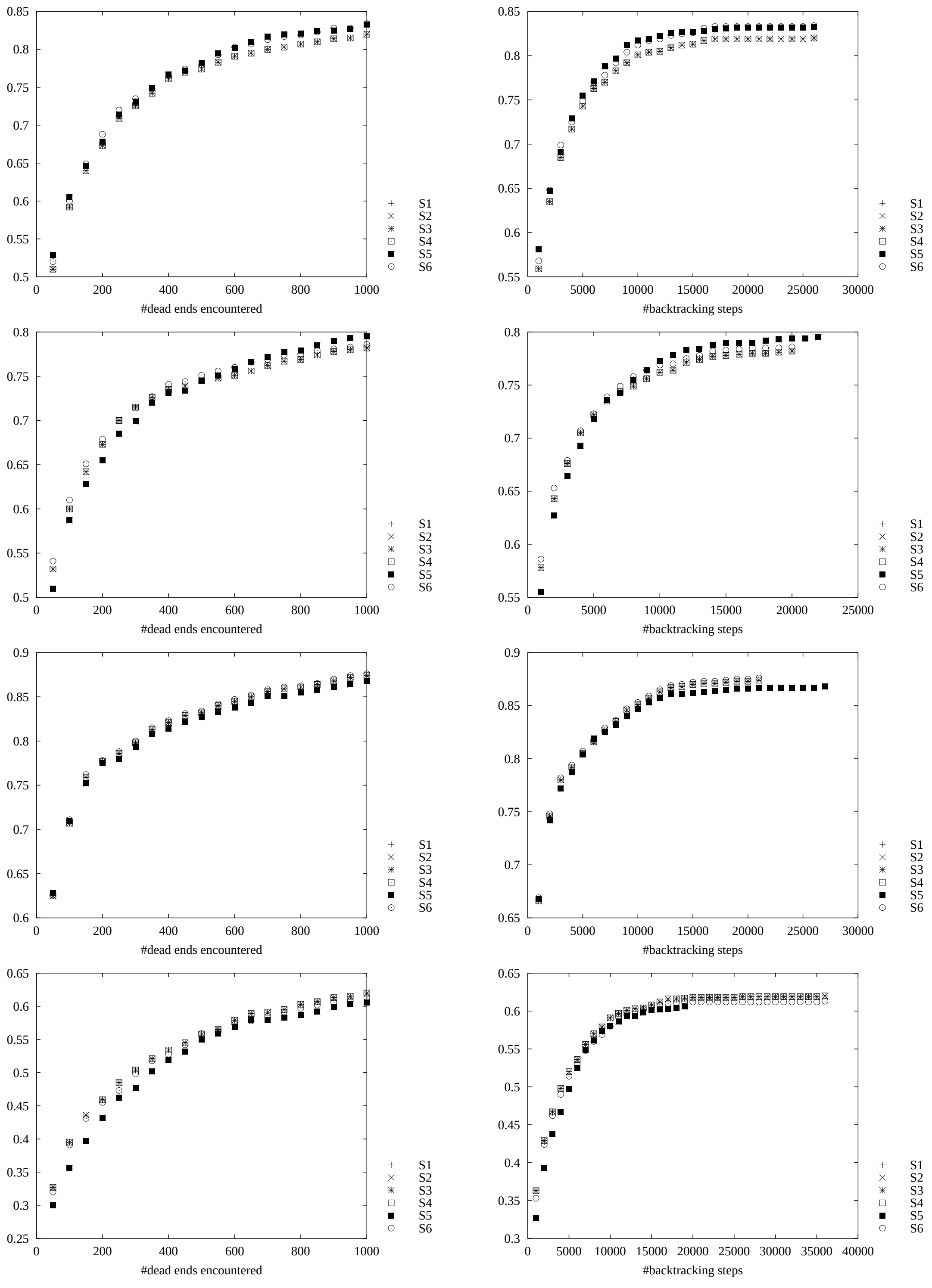


Figure 7: Behaviour of solvers for schools R3–R6 (from top to bottom)

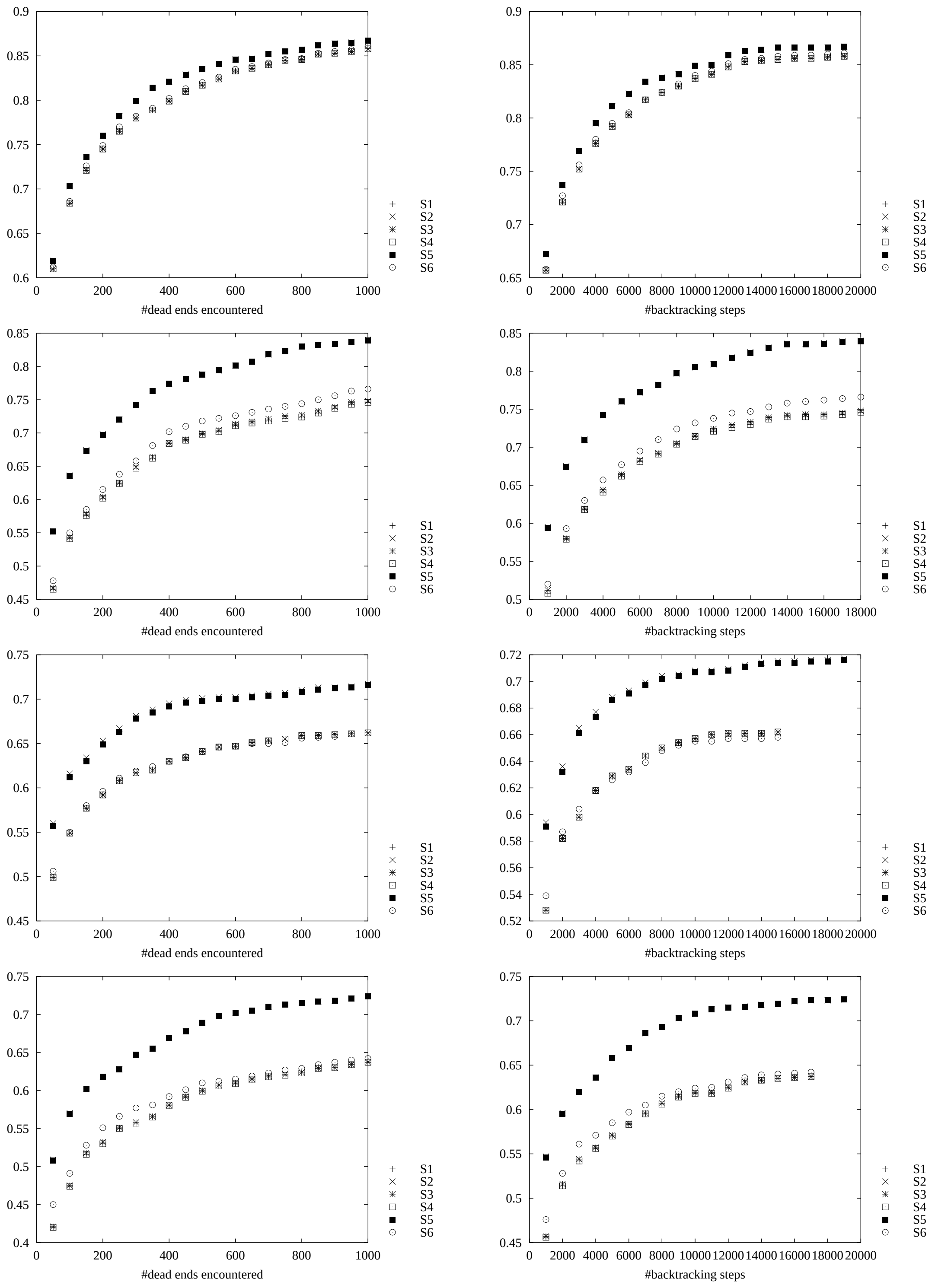


Figure 8: Behaviour of solvers for schools A1–A4 (from top to bottom)

References

[1] D. Appelt. *Education in Bavaria*. Bavarian State Ministry of Education and Cultural Affairs, 1998.

[2] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[3] N. Beldiceanu and M. Carlsson. Sweep as a generic pruning technique applied to the non-overlapping rectangles constraint. In T. Walsh, editor, *Seventh International Conference on Principles and Practice of Constraint Programming*, LNCS 2239, pages 377–391. Springer, 2001.

[4] P. Brucker and L. Nordmann. The *k*-track assignment problem. *Computing*, 52(2):97–122, 1994.

[5] 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.

[6] P. R. Cohen. *Empirical Methods for Artificial Intelligence*. MIT Press, 1995.

[7] A. Drexl and F. Salewski. Distribution requirements and compactness constraints in school timetabling. *European Journal of Operational Research*, 102(1):193–214, 1997.

[8] U. Faigle, W. Kern, and W. M. Nawijn. A greedy on-line algorithm for the *k*-track assignment problem. *Journal of Algorithms*, 31(1):196–210, 1999.

[9] Intelligent Systems Laboratory, Swedish Institute of Computer Science. *SICStus Prolog User's Manual, Release 3.9.0*, 2002.

[10] A. W. J. Kolen and J. K. Lenstra. Combinatorics in operations research. In R. L. Graham, M. Grötschel, and L. Lovácz, editors, *Handbook of Combinatorics*, volume 2. Elsevier, 1995.

[11] M. Marte. A modular approach to proving confluence. Technical Report PMS-FB-2001-18, Institut für Informatik der Universität München, 2001.

[12] 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.

[13] K. Mehlhorn and S. Näher. *LEDA: A Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1999.

[14] J.-C. Régin. A filtering algorithm for constraints of difference in CSPs. In *Proceedings of the 12th National Conference on Artificial Intelligence*, pages 362–367. AAAI Press, 1994.

[15] J.-C. Régin. Generalized arc consistency for global cardinality constraint. In *Proceedings of the 13th National Conference on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conference*, pages 209–215. AAAI Press, 1996.

[16] M. Sarrafzadeh and D. T. Lee. Restricted track assignment with applications. *International Journal of Computational Geometry and Applications*, 4(1):53–68, 1994.

[17] P. Van Hentenryck, V. Saraswat, and Y. Deville. Design, implementation, and evaluation of the constraint language cc(FD). *Journal of Logic Programming*, 37(2):139–164, 1998.

[18] W. J. van Hoeve. The alldifferent constraint: A survey. In *6th Annual Workshop of the ERCIM Working Group on Constraints*, 2001.