Precedence-constrained scheduling problems parameterized by partial order width

René van Bevern[1]

joint work with

Robert Bredereck[2], Laurent Bulteau[3], Christian Komusiewicz[4], Nimrod Talmon[5], Gerhard J. Woeginger[6]

[1]Novosibirsk State University and Sobolev Institute of Mathematics, Russian Federation,

[2]University of Oxford, United Kingdom [3]Université Paris-Est Marne-la-Vallée, France

[4]Friedrich-Schiller Universität Jena, Germany [5]Weizmann Institute of Science, Israel

[6]TU Eindhoven, The Netherlands

DOOR 2016, September 19–23, 2016, Vladivostok, Russian Federation

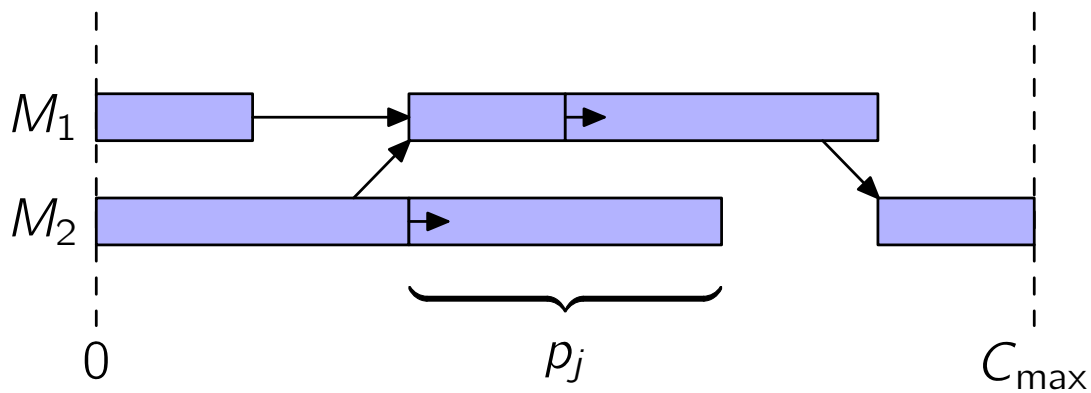
1 Introduction

Problem 1.1 (P|prec|$C_{\max}$)**.**

Input: A number m of machines, a set $\mathcal{J} = \{J_1, \ldots, J_n\}$ of jobs, each job J_j with a processing time p_j, and a partial order $\preceq$ on $\mathcal{J}$.

Task: Process all jobs on m parallel identical machines such that

1. each machine processes at most one job at a time,
2. each job is processed non-preemptively by exactly one machine,
3. for jobs $J_j \prec J_k$, job J_j is finished before J_k starts,
4. the maximum completion time $C_{\max}$ is minimized.

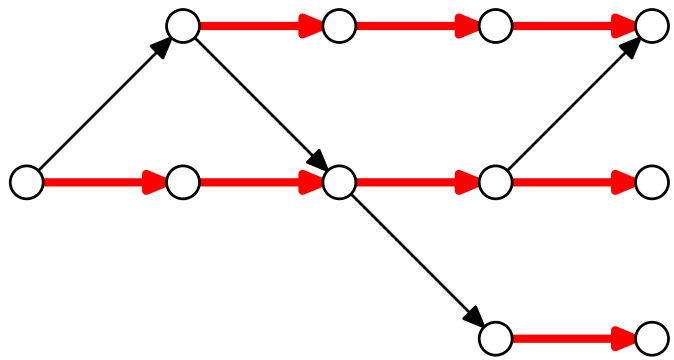


Mnich and Wiese, Math. Progr. 154(1–2):533–562, 2015:

Showed: $P||C_{\max}$ (i. e., without precedence constraints) is solvable in $f(p_{\max}) \cdot n^{O(1)}$ time, where $p_{\max}$ is the maximum processing time.

Asked: Is $P|\text{prec}|C_{\max}$ solvable in $f(p_{\max}, w) \cdot n^{O(1)}$ time?

Width w of a partial order $\preceq$: number of *chains* required to cover it:



Theorem 1.2. $P2|\text{prec}, p_j \in \{1,2\}|C_{\max}$ is not solvable in $f(w) \cdot n^{O(1)}$ time, unless FPT = W[2].[1]

[1] For $P|\text{prec}, p_j = 1|C_{\max}$ this was shown already in 1995 by Bodlaender and Fellows, Oper. Res. Lett. 18(2):93–97, answering Mnich and Wiese's question 20 years before posed.

2 Shuffle Products

Hardness of P2|prec,$p_j \in \{1,2\}$|$C_{\max}$ by resolving seemingly unrelated question:

Problem 2.1 (Shuffle Product)**.**

Input: Words $s_1, \ldots, s_k$ and t over some alphabet Σ.

Question: Can t be obtained by interleaving the letters of $s_1, \ldots, s_k$?

$s_1 =$	a			c		b			b
$s_2 =$			b		b		c		
$s_3 =$		c						a	b
$t =$	a	c	b	c	b	b	c	a	b	b

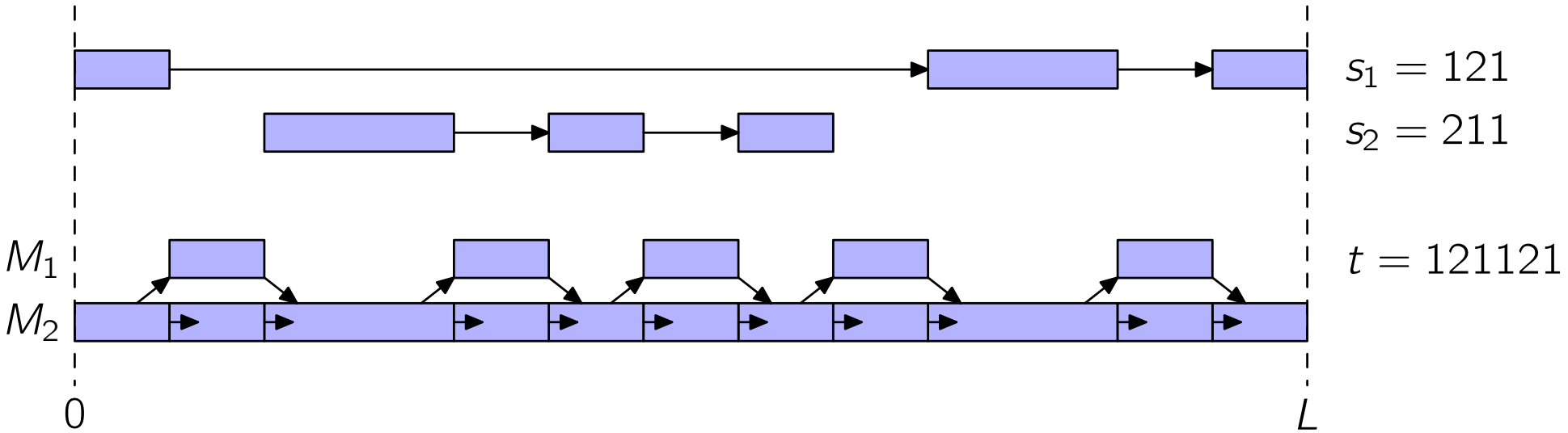
Rizzi and Vialette, CSR 2013: Shuffle Product is polynomial-time solvable for each constant k. Is it solvable in $f(k) \cdot n^{O(1)}$ time?

Theorem 2.2. Shuffle Product is not solvable in $f(k) \cdot n^{O(1)}$ time even if $|\Sigma| = 2$, unless FPT = W[2].

2.1 From shuffle products to scheduling

Theorem 2.2. P2|prec,$p_j \in \{1,2\}$|$C_{\max}$ is not solvable in $f(w) \cdot n^{O(1)}$ time, unless FPT = W[2].

Proof. Polynomial-time reduction from Shuffle Product with k strings over $\Sigma = \{1,2\}$ to P2|prec,$p_j \in \{1,2\}$|$C_{\max}$ with partial order $\preceq$ of width $w = k+2$.



Schedule has makespan L iff t can be obtained by interleaving s_1 and s_2. □

3 An algorithm for finding schedules with lag λ

Definition 3.1. Let $(\sigma_j)_{J_j \in \mathcal{J}}$ be the (not necessarily feasible) schedule that starts each job J_j at the earliest possible time σ_j, i. e.,

- $\forall J_j \in \mathcal{J}_0 : \sigma_j = 0$, where $\mathcal{J}_0$ are the jobs without predecessors,
- $\forall J_j \in \mathcal{J} \setminus \mathcal{J}_0 : \sigma_j = \max_{J_k \prec J_j}(\sigma_k + p_k)$.

A schedule $(s_j)_{J_j \in \mathcal{J}}$ that starts job J_j at time s_j has *lag at most* λ if

$$\forall J_j \in \mathcal{J} : s_j - \sigma_j \leq \lambda.$$

Theorem 3.2. $P|prec|C_{max}$ is solvable in $(2\lambda + 1)^w \cdot n^{O(1)}$ time, and so is the more general Resource-Constrained Project Scheduling Problem.

Refines pseudo-polynomial algorithm for Resource-Constrained Project Scheduling with constant width w (Servakh, Diskretn. Anal. Issled. Oper. 7(1):75–82).

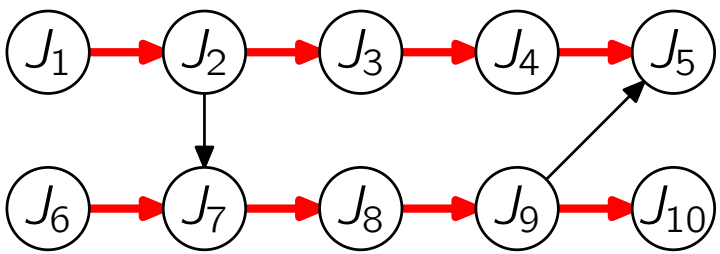
3.1 Based on geometric methods for Job Shop from the 50s:

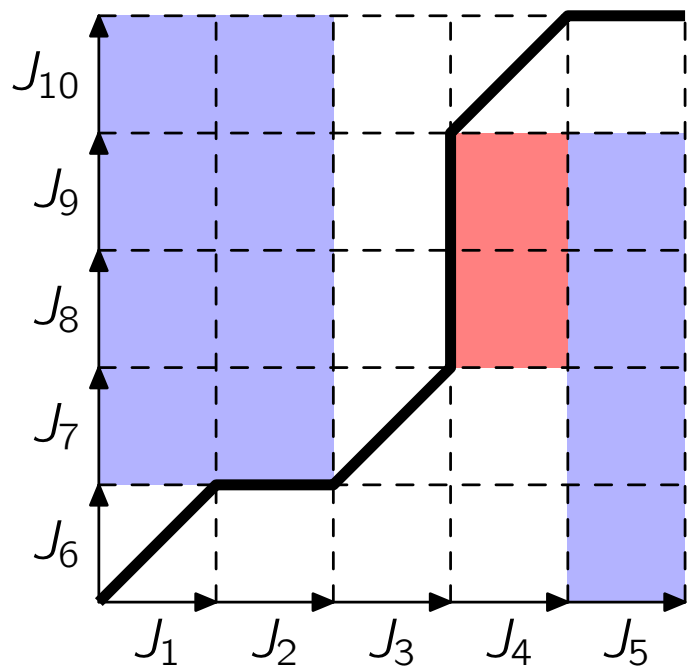
Optimal solution $\equiv$ shortest feasible path in w-dimensional orthotope with the w chains of $\preceq$ as axes. Compute it using dynamic programming.

Jobs that cannot be processed at the same time due to ...

... some resource constraints,

... precedence constraints:

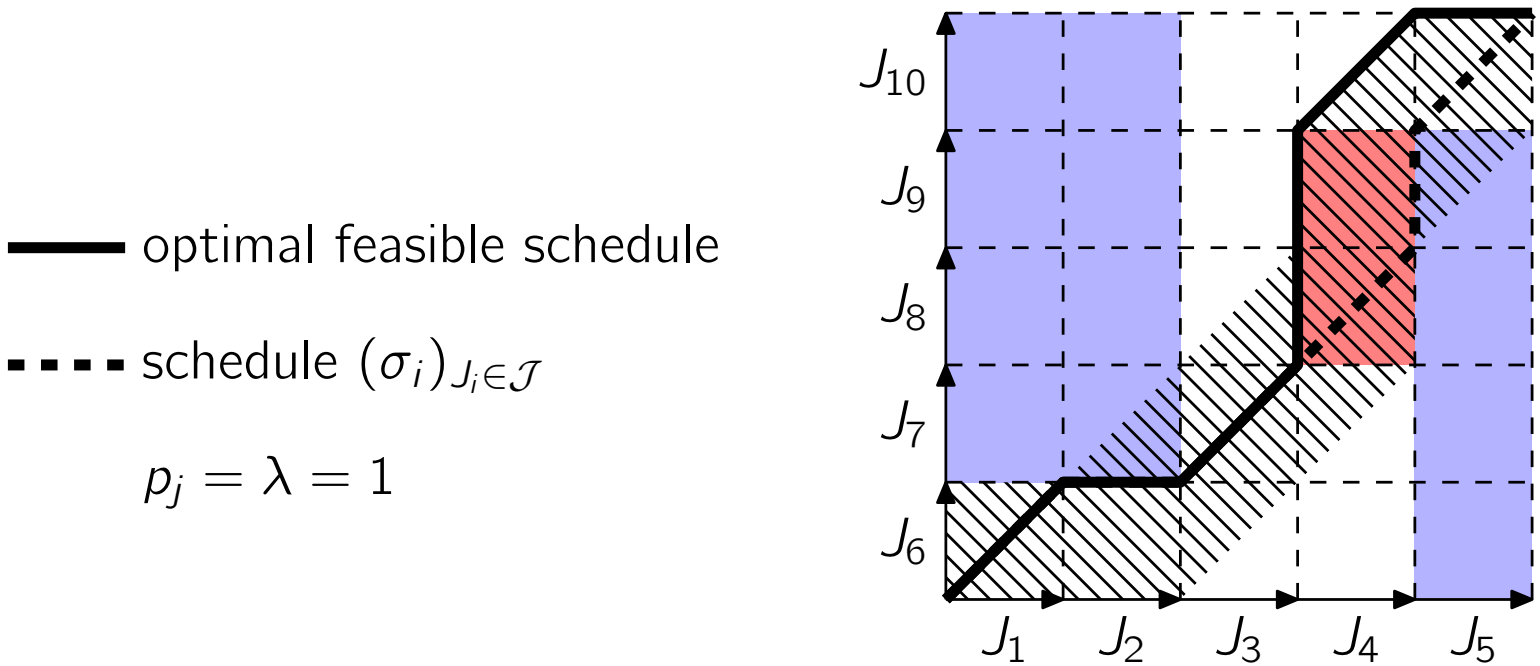




Takes as much time as there are integer points in the orthotope: $\prod_{\ell=1}^{w}(1+P_\ell)$, where P_ℓ is the sum of processing times on chain ℓ of $\preceq$.

3.2 Schedules of lag at most λ with pseudo-polynomial

Observation 3.3. Path of schedule with lag at most λ is within a λ-wide corridor around path of earliest-possible schedule $(\sigma_j)_{J_j \in \mathcal{J}}$.



Dynamic programming takes as much time as there are integer points in the corridor: at most $(\lambda + 1)^w \cdot P^{O(1)}$, where P is sum of all processing times.

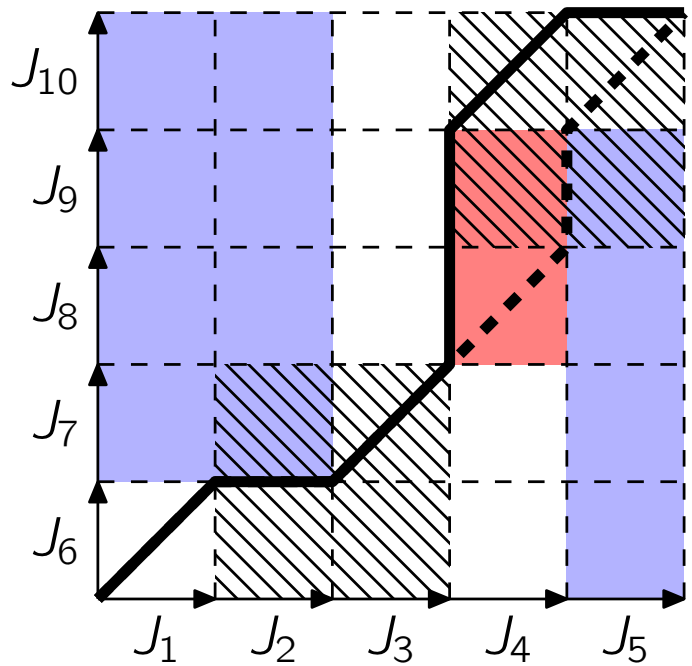
3.3 Schedules of lag at most λ with true polynomial

Observation 3.4. Path of schedule with lag at most λ changes direction only in $2n$ many w-dimensional hypercubes of edge-length 2λ each.

optimal feasible schedule

schedule $(\sigma_i)_{J_i \in \mathcal{J}}$

$p_j = \lambda = 1$



Dynamic programming takes as much time as there are integer points in these hypercubes: at most $(2\lambda + 1)^w \cdot n^{O(1)}$.

3.4 Is the exponential dependence on both λ and w necessary?

We have shown that:

- $P|prec|C_{max}$ (and Resource-Constrained Project Scheduling)
- with lag λ and width w is solvable in $(2\lambda+1)^w \cdot n^{O(1)}$ time.

This result is tight in the following sense:

- $P2|chains|C_{max}$ is (weakly) NP-hard even for constant $w = 3$.[2]
- $P|prec|C_{max}$ is (strongly) NP-hard for constant $\lambda = 1$.[3]

[2] Günther, König, Megow, J. Comb. Opt. 27(1):164–181, 2014

[3] Lenstra and Rinnooy Kan, Oper. Res. 26(1):22-35, 1978