Koutecký, Martin; Zink, Johannes

Complexity of scheduling few types of jobs on related and unrelated machines

Journal of Scheduling

Provided in Cooperation with:
Springer Nature

Suggested Citation: Koutecký, Martin; Zink, Johannes (2025) : Complexity of scheduling few types of jobs on related and unrelated machines, Journal of Scheduling, ISSN 1099-1425, Springer US, New York, NY, Vol. 28, Iss. 1, pp. 139-156, https://doi.org/10.1007/s10951-024-00827-8

Journal of Scheduling (2025) 28:139–156
https://doi.org/10.1007/s10951-024-00827-8

Complexity of scheduling few types of jobs on related and unrelated machines

Martin Koutecký[1] · **Johannes Zink**[2]

Accepted: 13 October 2024 / Published online: 30 January 2025

Abstract

The task of scheduling jobs to machines while minimizing the total makespan, the sum of weighted completion times, or a norm of the load vector are among the oldest and most fundamental tasks in combinatorial optimization. Since all of these problems are in general NP-hard, much attention has been given to the regime where there is only a small number k of job types, but possibly the number of jobs n is large; this is the few job types, high-multiplicity regime. Despite many positive results, the hardness boundary of this regime was not understood until now. We show that makespan minimization on uniformly related machines ($Q|HM|C_{\max}$) is NP-hard already with 6 job types, and that the related CUTTING STOCK problem is NP-hard already with 8 item types. For the more general unrelated machines model ($R|HM|C_{\max}$), we show that if the largest job size $p_{\max}$ or the number of jobs n is polynomially bounded in the instance size $|I|$, there are algorithms with complexity $|I|^{\text{poly}(k)}$. Our main result is that this is unlikely to be improved because $Q||C_{\max}$ is W[1]-hard parameterized by k already when n, $p_{\max}$, and the numbers describing the machine speeds are polynomial in $|I|$; the same holds for $R||C_{\max}$ (without machine speeds) when the job sizes matrix has rank 2. Our positive and negative results also extend to the objectives ℓ_2-norm minimization of the load vector and, partially, sum of weighted completion times $\sum w_j C_j$. Along the way, we answer affirmatively the question whether makespan minimization on identical machines ($P||C_{\max}$) is fixed-parameter tractable parameterized by k, extending our understanding of this fundamental problem. Together with our hardness results for $Q||C_{\max}$, this implies that the complexity of $P|HM|C_{\max}$ is the only remaining open case.

Keywords High-multiplicity jobs · Cutting stock · Hardness · Parameterized complexity

1 Introduction

Makespan minimization is arguably the most natural and most studied scheduling problem: In the parallel machines model, we have m machines, n jobs with sizes $p_1, \ldots, p_n$, and the task is to assign them to machines such that the sum of sizes of jobs on any machine is minimized. Seen differently, this is the (decision version of the) BIN PACKING problem: Can a set of items be packed into a given number of bins? BIN PACKING is NP-hard, so it is natural to ask which restrictions make it polynomial-time solvable. Say there are only k distinct item sizes $p_1, \ldots, p_k$, and so the items are given by a vector of multiplicities $n_1, \ldots, n_k$ with $n = \sum_{j=1}^{k} n_j$; let $p_{\max} = \max_{j \in [k]} p_j$. Goemans and Rothvoß (2020) and Jansen and Klein (2020) showed that BIN PACKING can be solved in time $(\log p_{\max})^{f(k)} \operatorname{poly} \log n$ for some function f,[1] Note that makespan minimization is polynomial when k is fixed by simple dynamic programming; the difficult question is whether it is still polynomial in the *high-multiplicity* setting where jobs are encoded by the multiplicity vector $\mathbf{n} = (n_1, \ldots, n_k)$. By the equivalence with scheduling, Goemans and Rothvoß showed that high-multiplicity makespan minimization on identical machines is polynomial if the number of job types k is fixed.

A preliminary version of this work has appeared in *Proceedings of the 31st International Symposium on Algorithms and Computation* (ISAAC 2020).

✉ Martin Koutecký
koutecky@iuuk.mff.cuni.cz

✉ Johannes Zink
zink@informatik.uni-wuerzburg.de

[1] Charles University, Prague, Czech Republic

[2] Universität Würzburg, Würzburg, Germany

[1] The complexity stated in Goemans and Rothvoß (2020) is $(\log \max\{C_{\max} n\})^{f(k)} \operatorname{poly} \log n$, but a close inspection of their proof reveals that the first dependence on n is unnecessary, and the work of Jansen and Klein (2020) allows bounding the complexity in terms of the number of vertices of a polytope, which can be independent of the right-hand side $C_{\max}$ (Berndt et al., 2021). This way, one can obtain the complexity stated here.

Since 2014 (when Goemans and Rothvoß published the conference version of their work (2014)), considerable attention has been given to studying the complexity of various scheduling problems in the regime with few job types (Jansen & Klein, 2020; Knop et al., 2019; Knop & Koutecký, 2018; Knop & Koutecký, 2022; Chen et al., 2017; Hermelin et al., 2021; Mnich & Wiese, 2015; Jansen, 2017; Hermelin et al., 2019, and similar techniques have been used to obtain approximation algorithms (Jansen et al., 2018; Levin, 2022; Jansen et al., 2020). However, any answer to the following simple and natural question was curiously missing:

> *What is the most restricted machine model in which high-multiplicity makespan minimization becomes* NP-hard, *even when the number of job types is fixed?*

There are three main machine models in scheduling: identical, uniformly related, and unrelated machines. In the uniformly related machines model, machine M_i (for $i \in [m]$) additionally has a *speed* s_i, and processing a job of size p_j takes time p_j/s_i on such a machine. In the unrelated machines model, each machine M_i (for $i \in [m]$) has its own vector of job sizes $\mathbf{p}^i = (p^i_1, \ldots, p^i_k)$, so that p^i_j is the time to process a job of type j on machine M_i. The makespan minimization problem in the identical, uniformly related, and unrelated machines model is denoted as $P||C_{\max}$, $Q||C_{\max}$, and $R||C_{\max}$ (Lawler et al., 1993), respectively, with the high-multiplicity variants being $P|HM|C_{\max}$, $Q|HM|C_{\max}$, and $R|HM|C_{\max}$. Notice that the job sizes matrix $\mathbf{p}$ of a $Q||C_{\max}$ instance is of rank 1: The vector $\mathbf{p}^i$ for machine M_i is simply $\mathbf{p}'/s_i$ for $\mathbf{p}' = (p_1, \ldots, p_k)$, and $\mathbf{p} = \mathbf{p}' \cdot (1/\mathbf{s})^\intercal$ for the speeds vector $\mathbf{s} = (s_1, \ldots, s_m)$ and its vector of reciprocal values $(1/\mathbf{s}) = (1/s_1, \ldots, 1/s_m)$. Hence, the rank of the job sizes matrix has been studied (Bhaskara et al., 2013; Chen et al., 2017, 2018) as a helpful measure of the complexity of an $R||C_{\max}$ instance: Intuitively, the smaller the rank, the closer is the instance to $Q||C_{\max}$.

Regarding the question above, it has been shown by McCormick et al. (2001) that for just 2 different job types, the problem $Q|HM|C_{\max}$ is polynomial-time solvable, but so far it remained open whether $Q|HM|C_{\max}$ becomes NP-hard for some constant number of job types at all—as $P|HM|C_{\max}$ remains polynomial-time solvable. We now close this gap for the most part:

Theorem 4.13 $Q|HM|C_{\max}$ is NP-hard already for 6 job types.

The Cutting Stock problem relates to Bin Packing in the same way as $Q||C_{\max}$ relates to $P||C_{\max}$: Instead of having all bins have the same capacity, there are now several bin types with different capacities and costs, and the task is to pack all items into bins of minimum cost. Cutting Stock is a famous and fundamental problem whose study dates back to the ground-breaking work of Gilmore and Gomory (1961) from 1961. It is thus surprising that the natural question whether Cutting Stock with a fixed number of item types is polynomial or NP-hard has not been answered until now:

Theorem 4.17 Cutting Stock is NP-hard already with 8 item types.

Parameterized Complexity.
A more precise complexity landscape can be obtained by taking the perspective of parameterized complexity: We say that a problem is *fixed-parameter tractable* (FPT, or *in* FPT, for short) parameterized by a *parameter* k if there is an algorithm solving any instance I in time $f(k)\,\mathrm{poly}(|I|)$ for some computable function f. On the other hand, showing that a problem is W[1]-hard means that it is unlikely to have such an algorithm, and the best one might hope for is a complexity of the form $|I|^{f(k)}$; we then say that a problem is *in* XP (or that it *has an XP algorithm*); see the textbook by Cygan et al. (2015) for more information on parameterized complexity.

The hardness instance I that we use to prove Theorem 4.13 is encoded by a job sizes matrix $\mathbf{p}$, a job multiplicities vector $\mathbf{n}$, and a machine speeds vector $\mathbf{s}$ which all contain long numbers, i.e., entries with encoding length $\Omega(|I|)$. What happens when some of $\mathbf{p}$, $\mathbf{n}$, and $\mathbf{s}$ are restricted to numbers bounded by $\mathrm{poly}(|I|)$, or, equivalently, if they are encoded in unary?

A note of caution: since we allow speeds to be rational, and the encoding length of a fraction p/q is $\lceil \log_2 p + \rceil \log_2 q$, a $Q||C_{\max}$ instance with $\mathbf{s}$ of polynomial size (with respect to the input length) might translate to an $R||C_{\max}$ instance with $\mathbf{p}$ of exponential size. This is because for $\mathbf{p}$ to be integer, one needs to scale it up by the least common multiple of the denominators in $\mathbf{s}$, which may be exponential in m. Thus, with respect to the magnitude of $\mathbf{n}$ and $\mathbf{p}$, $R|HM|C_{\max}$ can *not* be treated as a generalization of $Q|HM|C_{\max}$. This is why in the following, we deal with both problems and not just the seemingly more or less general one. For $Q|HM|C_{\max}$, we denote by $p_{\max}$ the largest job size *before scaling*, i.e., if $\mathbf{p} = \mathbf{p}' \cdot (1/\mathbf{s})^\intercal$, then $p_{\max} = \|\mathbf{p}'\|_\infty$.

Having $\mathbf{n}$ polynomially bounded is equivalent to giving each job explicitly; note that in this setting $R|HM|C_{\max}$ strictly generalizes $Q|HM|C_{\max}$. A simple DP handles this case:

Theorem 3.1 $\{R, Q\}|HM|C_{\max}$ and $\{R, Q\}||C_{\max}$ can be solved in time $m \cdot n^{\mathcal{O}(k)}$, hence $\{R, Q\}||C_{\max}$ is in XP parameterized by k.

A similar situation occurs if $\mathbf{n}$ is allowed to be large, but $\mathbf{p}$ is polynomially bounded, although the use of certain integer programming tools (Eisenbrand et al., 2019) is required:

Theorem 3.7 $\{R, Q\}|HM|C_{\max}$ can be solved in time $p_{\max}^{\mathcal{O}(k^2)} m \log m \log^2 n$, and hence, $\{R, Q\}|HM|C_{\max}$ are in XP parameterized by k if $p_{\max}$ is given in unary.

Our main result is that an FPT algorithm for $Q|HM|C_{\max}$ is unlikely to exist even when **n**, **p**, and **s** are encoded in unary, and for $R|HM|C_{\max}$ even when the rank of **p** is 2:

Theorem 4.15 *$X||C_{\max}$ is W[1]-hard parameterized by the number of job types with (a) $X = Q$ and even when **n**, **p**, and **s** are given in unary; (b) $X = R$ and even when **n** and **p** are given in unary and $\text{rank}(\mathbf{p}) = 2$.*

We use a result of Jansen et al. (2013) as the basis of our hardness reduction. They show that Bin Packing is W[1]-hard parameterized by the number of bins even if the items are numbers given in unary coding. In the context of scheduling, this means that $P||C_{\max}$ is W[1]-hard parameterized by the number of machines already when $p_{\max}$ is polynomially bounded. However, it is non-obvious how to "transpose" the parameters, that is, how to go from many job types and few machines to few job types and many machines which differ as little as possible (i.e., only by their speeds, or only in a low-rank way). We first show W[1]-hardness of Balanced Bin Packing, where we additionally require that the number of items in each bin is identical, parameterized by the number of bins, even for tight instances in which each bin has to be full. Using this additional property, we are able to construct a $Q|HM|C_{\max}$ instance of makespan T in which optimal solutions are in bijection with optimal packings of the encoded Balanced Bin Packing instance. For each item in the Balanced Bin Packing instance, we have a machine in the instance of $Q|HM|C_{\max}$, whose speed depends on the item the machine represents. We can transform this also to a hardness instance of $R|HM|C_{\max}$, where we do not have individual speeds for the machines, by using a new machine-dependent job type to "block out" a large part of a machine's capacity so that its remaining capacity depends on the item the machine represents. Then, all other job types have sizes independent of which machine they run on as for the hardness instance of $Q|HM|C_{\max}$.

Let us go back to $P|HM|C_{\max}$. As mentioned previously, Goemans and Rothvoß showed that if the largest job size $p_{\max}$ is polynomially bounded, the problem is FPT because $(\log p_{\max})^{f(k)} \text{ poly} \log n \leq g(k) \cdot p_{\max}^{o(1)} \text{ poly} \log n$ (Cygan et al., 2015, Exercise 3.18). We answer the remaining question whether the problem is in FPT also when all jobs are given explicitly:

Theorem 3.5 *$P||C_{\max}$ is FPT parameterized by k.*

This result partially answers (Mnich and Bevern, 2018, Question 5), which asks for an FPT algorithm for $P|HM|C_{\max}$. Obtaining this (partial) answer turns out to be surprisingly easy: We reduce the job sizes by a famous algorithm of Frank and Tardos (1987) and then apply the algorithm of Goemans and Rothvoß (2020), which is possible precisely when n is sufficiently small. This extends our understanding of the complexity of $P|HM|C_{\max}$: The problem is FPT if the size of the largest job or the number of jobs are polynomially bounded in the input length. Hence, the remaining (and major) open problem is the complexity of $P|HM|C_{\max}$ parameterized by k, without any further assumptions on the magnitude of $p_{\max}$ or n. In light of this, our result that already $Q|HM|C_{\max}$ is NP-hard when $p_{\max}$ and n are large, and W[1]-hard if both are polynomially bounded, may be interpreted as indication that the magnitude of n and $p_{\max}$ plays a surprisingly important role, and that $P|HM|C_{\max}$ may in fact *not* be FPT parameterized by k.

Other Objectives.

Besides minimum makespan, two important scheduling objectives are minimization of the sum of weighted completion times, denoted by $\sum w_j C_j$, and the minimization of the ℓ_2-norm of the load vector. We show that our algorithms and hardness results (almost always) translate to these objectives as well. Let us now introduce them formally.

The load L_i of a machine M_i is the total size of jobs assigned to it. In $R|HM|\ell_2$, the task is to find a schedule minimizing $\|(L_1, \ldots, L_m)\|_2 = \sqrt{\sum_{i=1}^{m} L_i^2}$. Note that this is isotonic (order preserving) to the function $\sum_{i=1}^{m} L_i^2$, and because this leads to simpler proofs, we instead study the problem $R|HM|\ell_2^2$. The completion time of a job j, denoted by C_j, is the time it finishes its execution in a schedule. In the $R|HM|\sum w_j C_j$ problem, each job is additionally given a weight w_j, and the task is to minimize $\sum w_j C_j$.

We show that our hardness instance for $R|HM|C_{\max}$ is also hard for ℓ_2, and with the right choice of weights is also hard for $\sum w_j C_j$. We also obtain hardness of $Q|HM|\ell_2$ by a different and more involved choice of the speeds, but the case of $Q|HM|\sum w_j C_j$ remains open so far. To extend the $C_{\max}$ reduction to other objectives, we use the "tightness" of our hardness instance to show that any "non-tight" schedule must increase the ℓ_2 norm of the load vector by at least some amount. This is not enough for $R|HM|\sum w_j C_j$ because the value $\sum w_j C_j$ is proportional to the load vector plus other terms, and we need to bound those remaining terms (Lemma 4.23) in order to transfer the argument from ℓ_2 to $\sum w_j C_j$. We point out that the these hardness results are delicate and non-trivial even if at first sight they may appear as "just" modifying the hardness instance of $Q|HM|C_{\max}$.

We give an overview of our results in Table 1.

2 Preliminaries

We consider zero a natural number, i.e., $0 \in \mathbb{N}$. We write vectors in boldface (e.g., $\mathbf{x}, \mathbf{y}$) and their entries in normal font (e.g., the i-th entry of a vector $\mathbf{x}$ is x_i). If it is clear from context that $\mathbf{x}^\intercal\mathbf{y}$ is a dot-product of $\mathbf{x}$ and $\mathbf{y}$, we just write $\mathbf{xy}$ (Conforti et al., 2014). When we say *a vector is given in*

Table 1 Overview of the computational hardness of $\{P, Q, R\}|\{\_, HM\}|\{C_{\max}, \ell_2, \sum w_j C_j\}$ relative to the number of job types k. The cells are colored as follows. Gray are positive results known before, green are new positive results, red are new negative results, and yellow are the settings where the computational complexity is unknown

	$P\|\|\ldots$		$Q\|\|\ldots$		$R\|\|\ldots$	$P\|HM\|\ldots$	$Q\|HM\|\ldots$	$R\|HM\|\ldots$
$C_{\max}$	FPT (Thm. 3.5)	XP (Theorem 3.1)	W[1]-hard (Thm. 4.15)	XP (Theorem 3.1)	W[1]-hard (Thm. 4.15)	poly. time for const. k ([1])	poly. time for $k \leq 2$ ([20]), NP-hard for $k \geq 6$ (Thm. 4.13)	NP-hard for $k \geq 4$ (Thm. 4.14)
ℓ_2	?		W[1]-hard (Cor. 4.22)		W[1]-hard (Cor. 4.22)	?	NP-hard for $k \geq 6$ (Cor. 4.21)	NP-hard for $k \geq 7$ (Cor. 4.21)
$\sum w_j C_j$	?		?		W[1]-hard (Cor. 4.26)	?	?	NP-hard for $k \geq 7$ (Cor. 4.25)

unary, it means that each number in this vector is given in unary coding. We use $\log := \log_2$, i.e., all our logarithms are base 2. For $n, m \in \mathbb{N}$, we write $[n, m] = \{n, n+1, \ldots, m\}$ and $[n] = [1, n]$.

We first study the problem to minimize the makespan on uniformly related machines:

MAKESPAN MINIMIZATION ON UNIFORMLY RELATED MACHINES ($Q|HM|C_{\max}$)

Input: n jobs of k types, job multiplicities $n_1, \ldots, n_k$, i.e., $n_1 + \cdots + n_k = n$ and n_j is the number of jobs of type j, m machines, for each $i \in [m]$ a speed $s_i \in \mathbb{Q}_+$ such that processing a job of type j on machine M_i takes time p_j/s_i, a number T.

Find: An assignment of jobs to machines and non-overlapping (with respect to each machine) time slots such that every machine finishes by time T.

We are also interested in a generalization of $Q||C_{\max}$ in which machines may be incomparable, e.g., when different machine types are more suitable for different job types:

MAKESPAN MINIMIZATION ON UNRELATED MACHINES ($R|HM|C_{\max}$)

Input: n jobs of k types, job multiplicities $n_1, \ldots, n_k$, i.e., $n_1 + \cdots + n_k = n$ and n_j is the number of jobs of type j, m unrelated machines, for each $i \in [m]$ a job sizes vector $\mathbf{p}^i = (p^i_1, \ldots, p^i_k) \in (\mathbb{N} \cup \{+\infty\})^k$, where p^i_j is the processing time of a job of type j on a machine M_i, a number T.

Find: An assignment of jobs to machines and non-overlapping (with respect to each machine) time slots such that every machine finishes by time T.

Note that $R|HM|C_{\max}$ generalizes $Q|HM|C_{\max}$ because we might take $\mathbf{p}^i = (p_j/s_i)_j$. However, the entries of $\mathbf{p}^i$ might then be fractional, so in order for this to correspond to the definition above, we would need to scale all $\mathbf{p}^i$ up by the least common multiple of all the appearing denominators. This would not increase the encoding length of the instance, but it might blow up the parameter $p_{\max}$, so it is indeed better to treat $Q|HM|C_{\max}$ and $R|HM|C_{\max}$ as distinct problems.

Also, notice that our definition uses a *high-multiplicity* encoding of the input, that is, jobs are not given explicitly, one by one, but "in bulk" by a vector of multiplicities. Because this allows compactly encoding instances which would otherwise be of exponential size, the two problems actually have different complexities and deserve a notational distinction: $R||C_{\max}$ the problem where jobs are given explicitly, and by $R|HM|C_{\max}$ the problem defined above; see also the discussion in Knop and Koutecký (2022).

Recall that in $R|HM|\ell_2$, the task is to minimize $\|(L_1, \ldots, L_m)\|_2$, where L_i is the sum of sizes of jobs assigned to machine M_i for $i \in [m]$. In $R|HM|\sum w_j C_j$, each job j has a weight w_j, and a schedule determines the completion time C_j of that job. The task is then to minimize $\sum w_j C_j$.

The job sizes matrix $\mathbf{p} \in \mathbb{R}_+^{k \times m}$ has rank r if it can be written as a product of matrices $C \in \mathbb{R}^{k \times r}$ and $D \in \mathbb{R}^{r \times m}$. For example, as we have mentioned, in $Q||C_{\max}$, each machine has a speed $s_i \in \mathbb{R}_+$, and $\mathbf{p}^i = \mathbf{p}'/s_i$ for some $\mathbf{p}' \in \mathbb{N}^k$, so $\mathbf{p} = \mathbf{p}'(1/\mathbf{s})^\intercal$, where $\mathbf{s} = (s_1, \ldots, s_m)$, and hence, $\mathbf{p}$ has rank 1.

In the *identical machines* model, $\mathbf{p}^i = \mathbf{p}$ for all $i \in [m]$, and we denote it by $P||C_{\max}$. Its decision variant $P||C_{\max}$ is equivalent to BIN PACKING:

BIN PACKING
Input: n items of sizes $a_1, \ldots, a_n$, k bins, each with capacity B.
Find: An assignment of items to bins such that the total size of items in each bin is $\leq B$.

UNARY BIN PACKING is BIN PACKING where the numbers $a_1, \ldots, a_n$ are encoded in unary each, or, equivalently, $a_{\max} = \max_{i \in [n]} a_i$ is bounded polynomially in n. BALANCED BIN PACKING is BIN PACKING with the additional requirement on the solution that the number of items assigned to each bin is the same, hence n/k; note that n has to be divisible by k for any instance to be feasible. An instance of BIN PACKING is *tight* if the total size of items $\sum_{i \in [n]} a_i$ is equal to $k \cdot B$, which means that if an instance has a packing, then each bin needs to be full.

3 Algorithms

We wish to highlight the geometric structure of $R|HM|C_{\max}$ by formulating it as an ILP and making several observations about it. We have a variable x_j^i for each job type $j \in [k]$ and machine M_i (with $i \in [m]$) specifying how many jobs of type j are scheduled to run on machine M_i. There are two types of constraints, besides the obvious bounds $\mathbf{0} \leq \mathbf{x}^i \leq \mathbf{n}$ for each $i \in [m]$. The first enforces that each job is scheduled somewhere, and the second assures that the sum of job sizes on each machine is at most T, meaning each machine finishes by time T:

$$\sum_{i=1}^{m} x_j^i = n_j \qquad \forall j \in [k] \tag{1}$$

$$\sum_{j=1}^{k} x_j^i p_j^i \leq T \qquad \forall i \in [m]\ . \tag{2}$$

Knop and Koutecký (2018) show that this ILP has *N-fold* format, i.e., it has the general form:

$$\min f(\mathbf{x}) : E^{(N)}\mathbf{x} = \mathbf{b},\ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u},\ \mathbf{x} \in \mathbb{Z}^{Nt},$$

$$\text{with } E^{(N)} = \begin{pmatrix} E_1^1 & E_1^2 & \cdots & E_1^N \\ E_2^1 & 0 & \cdots & 0 \\ 0 & E_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & E_2^N \end{pmatrix}.$$

Here, $r, s, t, N \in \mathbb{N}$, $E^{(N)}$ is an $(r + Ns) \times Nt$-matrix, for all $i \in [N]$, $E_1^i \in \mathbb{Z}^{r \times t}$ and $E_2^i \in \mathbb{Z}^{s \times t}$ are integer matrices, and f is some separable convex function. Specifically for $R||C_{\max}$, $f \equiv 0$, the matrices corresponding to constraints (1)–(2) are $E_1^i = I$ and $E_2^i = \mathbf{p}^i$, for each $i \in [m]$, $\mathbf{b} = (\mathbf{n}, T, \ldots, T)$ is an $r + Ns = (k + m)$-dimensional vector, and $\mathbf{l} = \mathbf{0}$ and $\mathbf{u} = (\mathbf{n}, \mathbf{n}, \ldots, \mathbf{n})$ are $Nt = (mk)$-dimensional vectors. (Actually, to get constraints (2) into the equality form $E^{(N)}\mathbf{x} = \mathbf{b}$, we need to introduce dummy jobs of size 1; it is straightforward to compute how many such jobs need to be introduced, for details see Knop and Koutecký (2018).) We note that N-fold IP formulations are also known for $R|HM|\{\ell_2, \sum w_j C_j\}$ (Knop and Koutecký, 2018; 2022).

3.1 Large lengths and polynomial multiplicities

A simple dynamic programming algorithm gives:

Theorem 3.1 *$\{R, Q\}|HM|\{C_{\max}, \ell_2, \sum w_j C_j\}$ can be solved in time $m \cdot n^{\mathcal{O}(k)}$, hence $\{R, Q\}||\{C_{\max}, \ell_2, \sum w_j C_j\}$ are in XP parameterized by k.*

Proof We will describe a simple dynamic programming (DP) algorithm. Call a vector $\mathbf{x}^i \in \mathbb{N}^k$ satisfying constraint (2), i.e., $\mathbf{p}^i \mathbf{x}^i \leq T$, a *configuration of machine M_i*. We will construct a DP table D indexed by k-dimensional integer vectors upper bounded by $\mathbf{n}$, and $i \in [m]$, and each value of the table is a 0/1 bit. The intended meaning is that, for $i \in [m]$ and $\mathbf{n}' \leq \mathbf{n}$, $D[i, \mathbf{n}'] = 1$ iff the subinstance consisting of jobs $\mathbf{n}'$ and the first i machines is feasible. Initialize D to be all-zero, and set $D[0, \mathbf{0}] = 1$. Then, consecutively for $i = 1, \ldots, m$, and for each $\mathbf{0} \leq \mathbf{n}' \leq \mathbf{n}$, set $D[i, \mathbf{n}'] = 1$ if $D[i-1, \mathbf{n}' - \mathbf{x}^i] = 1$ and $\mathbf{x}^i$ is a configuration of machine M_i. In other words, for each $i = 1, \ldots, m$, construct the set $\mathcal{C}^i$ of configurations of machine M_i, and then, for each $\mathbf{n}'$ with $D[i-1, \mathbf{n}'] = 1$, set $D[i, \mathbf{n}' + \mathbf{x}^i] = 1$ for each $\mathbf{x}^i \in \mathcal{C}^i$ if $\mathbf{n}' + \mathbf{x}^i \leq \mathbf{n}$. Finally, the instance is feasible if $D[m, \mathbf{n}] = 1$. In each iteration, we go over all $\mathbf{n}' \leq \mathbf{n}$, of which there is at most n^k many, and for each of them, we try to add each element of $\mathcal{C}^i$, of which there is also at most n^k many. In total, the algorithm makes $m \cdot n^k \cdot n^k = m \cdot n^{2k}$ steps.

The adaptation of this DP to ℓ_2 and $\sum w_j C_j$ is straightforward. Say that a configuration is any vector $\mathbf{x}^i \leq \mathbf{n}$. The value of a configuration $\mathbf{x}^i$ on machine M_i is $f^i(\mathbf{x}^i) = (\mathbf{p}^i \mathbf{x}^i)^2$ for ℓ_2^2. For $\sum w_j C_j$, it has been shown Knop & Koutecký (2018) that the contribution of a machine M_i scheduling jobs $\mathbf{x}^i$ is a quadratic convex function f^i in terms of $\mathbf{x}^i$. Then, $D[i, \mathbf{n}'] = \min_{\mathbf{x}^i \leq \mathbf{n} - \mathbf{n}'} f^i(\mathbf{x}^i) + D[i-1, \mathbf{n}' - \mathbf{x}^i]$. □

Theorem 3.1 (with a worse complexity bound) can be also shown in a somewhat roundabout way by manipulating the ILP formulation (1)–(2). This approach will eventually give us the result that $P||C_{\max}$ is FPT parameterized by k. We use a famous result of Frank and Tardos (1987) which intuitively states that a hyperplane determined by an arbitrary (rational) normal vector $\mathbf{w}$ can be replaced by another hyperplane determined by a normal vector $\mathbf{w}'$ of bounded encoding

length such that it separates exactly the same integer points within some prescribed box. In other words, the coefficients of the hyperplane $\mathbf{w}$ can be reduced (replaced with smaller ones) while preserving certain geometric meaning; thus, the technique is called "coefficient reduction."

Proposition 3.2 (Frank & Tardos, 1987) *Given a rational vector $\mathbf{w} \in \mathbb{Q}^d$ and an integer M, there is a strongly polynomial algorithm which finds a $\bar{\mathbf{w}} \in \mathbb{Z}^d$ such that for every integer point $\mathbf{x} \in [-M, M]^d$, we have $\mathbf{wx} \geq 0 \Leftrightarrow \bar{\mathbf{w}}\mathbf{x} \geq 0$ and $\|\bar{\mathbf{w}}\|_\infty \leq 2^{\mathcal{O}(d^3)} M^{\mathcal{O}(d^2)}$.*

Lemma 3.3 *It is possible to compute in strongly polynomial time for each $i \in [m]$ a vector $\bar{\mathbf{p}}^i \in \mathbb{N}^k$ and an integer $\bar{T}^i \in \mathbb{N}$ such that replacing constraint (2) with $\bar{\mathbf{p}}^i \mathbf{x}^i \leq \bar{T}^i$ does not change the set of feasible integer solutions, and $\|\bar{\mathbf{p}}^i, \bar{T}^i\|_\infty \leq 2^{\mathcal{O}(k^3)} n^{\mathcal{O}(k^2)}$.*

Proof Fix some $i \in [m]$ and consider the inequality (2), which is $\mathbf{p}^i \mathbf{x}^i \leq T$. Applying Proposition 3.2 to $(\mathbf{p}^i, T)$ and $M = n$ gives a vector $(\bar{\mathbf{p}}^i, \bar{T}^i)$ such that for all $\mathbf{0} \leq \mathbf{x}^i \leq \mathbf{n}$,

$$(\mathbf{p}^i, T)(\mathbf{x}^i, -1) \leq 0 \Leftrightarrow (\bar{\mathbf{p}}^i, \bar{T}^i)(\mathbf{x}^i, -1) \leq 0,$$

which means that replacing $\mathbf{p}^i \mathbf{x}^i \leq T$ by $\bar{\mathbf{p}}^i \mathbf{x}^i \leq \bar{T}$ in (2) does not change the set of feasible solutions, and the bound on $\|\bar{\mathbf{p}}^i, \bar{T}\|_\infty$ follows immediately from Proposition 3.2. □

We will use the fact that an N-fold IP can be solved efficiently:

Proposition 3.4 (Jansen et al., 2020; Cslovjecsek et al., 2020; Eisenbrand et al., 2019) *A feasibility instance of N-fold IP can be solved in time*

$$(\|E^{(N)}\|_\infty rs)^{\mathcal{O}(r^2 s + s^2)} Nt \log(Nt) \log^2 \|\mathbf{u} - \mathbf{l}\|_\infty .$$

Alternative proof of Theorem 3.1 for $C_{\max}$

According to Lemma 3.3, we can reduce $\|E^{(N)}\|_\infty$ down to $2^{\mathcal{O}(k^3)} n^{\mathcal{O}(k^2)}$. Since $r = k$, $t = k$, $s = 1$, $N = m$, and $\|\mathbf{u} - \mathbf{l}\|_\infty \leq n$, applying Proposition 3.4 to such a reduced instance gives an $n^{\mathcal{O}(k^5)} m \log m \log^2 n$ algorithm. Dealing with ℓ_2 and $\sum w_j C_j$ is analogous, see Lemma 3.6. □

While this is worse than the DP above, notice that this approach also gives:

Theorem 3.5 *$P||C_{\max}$ is FPT parameterized by k.*

Proof Apply Lemma 3.3 to a given $P||C_{\max}$ instance, which gives a new job sizes vector $\bar{\mathbf{p}} \in \mathbb{N}^k$ and a new time bound $\bar{T} \in \mathbb{N}$. Goemans and Rothvoß (2020) have shown that $P||C_{\max}$ with k job types can be solved in time $(\log p_{\max})^{2^{\mathcal{O}(k)}}$ poly $\log n$. Plugging in $p_{\max} \leq 2^{\mathcal{O}(k^3)} n^{\mathcal{O}(k^2)}$ gives $\log p_{\max} \leq \log(2^{\mathcal{O}(k^3)} n^{\mathcal{O}(k^2)}) = k^3 + k^2 \log n$. Hence, the algorithm runs in time $(k^3 \log n)^{2^{\mathcal{O}(k)}} = (k^3)^{2^{\mathcal{O}(k)}} \cdot (\log n)^{2^{\mathcal{O}(k)}}$. To verify that this is indeed an FPT runtime (i.e., $f(k)\,\text{poly}(n)$ for some computable f), we use a simple observation (Cygan et al., 2015, Exercise 3.18) that $(\log \alpha)^\beta \leq 2^{\beta^2/2} \alpha^{o(1)}$. Taking $\alpha = n$ and $\beta = 2^{\mathcal{O}(k)}$ gives $(\log n)^{2^{\mathcal{O}(k)}} \leq 2^{2^{\mathcal{O}(k)}} n^{o(1)}$, and we are done.

Remark 1 The algorithm of Goemans and Rothvoß (2020) shows that $P|HM|C_{\max}$ is FPT in k if $p_{\max}$ is given in unary. To the best of our knowledge, it has not been observed before that $P|HM|C_{\max}$ is FPT in k if n is polynomially bounded by the input length, i.e., that $P||C_{\max}$ is FPT in k. Thus, Theorem 3.5 shows that the remaining (and indeed hard) open problem is the complexity of $P|HM|C_{\max}$ for instances where both $\mathbf{p}$ and $\mathbf{n}$ contain large numbers.

The right-hand side $\mathbf{b}$ of an N-fold IP can be naturally partitioned into $N+1$ smaller vectors as $\mathbf{b} = (\mathbf{b}^0, \mathbf{b}^1, \ldots, \mathbf{b}^N)$ with $\mathbf{b}^0 \in \mathbb{Z}^r$ and $\mathbf{b}^i \in \mathbb{Z}^s$ for all $i \in [N]$. A straightforward adaptation of the proof of Lemma 3.3 where we reduce each row of the constraint $E_2^i \mathbf{x}^i = \mathbf{b}^i$ separately gives the following more general statement:

Lemma 3.6 *Given an N-fold IP instance with $\mathbf{0} \in [\mathbf{l}, \mathbf{u}]$, and $M \in \mathbb{N}$, one can in strongly polynomial time compute $\bar{E}_2^i$ and $\bar{\mathbf{b}}^i$, for each $i \in [N]$, such that if $\|\mathbf{u} - \mathbf{l}\|_\infty \leq 2M$, then*

$$\{\mathbf{x} \in \mathbb{Z}^{Nt} \mid E^{(N)}\mathbf{x} = \mathbf{b},\ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\} = \{\mathbf{x} \in \mathbb{Z}^{Nt} \mid \bar{E}^{(N)}\mathbf{x} = \bar{\mathbf{b}},\ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\},$$

where $\bar{E}^{(N)}$ is obtained from $E^{(N)}$ by replacing E_2^i with $\bar{E}_2^i$ and $\bar{\mathbf{b}}$ is obtained from $\mathbf{b}$ by replacing $\mathbf{b}^i$ with $\bar{\mathbf{b}}^i$, for each $i \in [N]$, and $\|\bar{E}_2^i, \bar{\mathbf{b}}^i\|_\infty \leq 2^{\mathcal{O}(t^3)} M^{\mathcal{O}(t^2)}$. □

Remark 2 If the assumption $\mathbf{0} \in [\mathbf{l}, \mathbf{u}]$ of the Lemma is not satisfied, it no longer holds. However, if $\|\mathbf{u} - \mathbf{l}\|_\infty \leq M$, it is still possible to transform the original instance into a new instance with small bounds and right-hand sides whose optimum is a simple translation of the original optimum. Thus, from an algorithmic perspective, the assumption $\mathbf{0} \in [\mathbf{l}, \mathbf{u}]$ is not necessary. Regarding how to translate an instance with $\mathbf{0} \notin [\mathbf{l}, \mathbf{u}]$, see (Eisenbrand et al., 2019, Lemma 50).

3.2 Polynomial lengths and large multiplicities

How to deal with instances whose jobs have polynomially bounded sizes, but come in large multiplicities? Actually, the fact that $R|HM|C_{\max}$ belongs to XP parameterized by k if $p_{\max}$ is polynomially bounded follows by solving the N-fold IP (1)–(2) using Proposition 3.4:

Theorem 3.7 *$\{R, Q\}|HM|\{C_{\max}, \ell_2, \sum w_j C_j\}$ and $\{R, Q\}|HM|\{\sum w_j C_j\}$ can be solved in time $p_{\max}^{\mathcal{O}(k^2)} m \log m \log^2 n$ and $p_{\max}^{\mathcal{O}(k^3)} m \log m \log^2 n$, respectively.*

An intuitive description of (a somewhat slower) algorithm is the following; for a formal but somewhat uninformative proof, see below. First, one would solve the LP relaxation of (1)–(2) and then use a "proximity theorem" to show that some integral optimum is at distance at most $p_{\max}^{\mathcal{O}(k)} \cdot m$ (Eisenbrand et al., 2019, Theorem 59) from any optimum of the LP relaxation. This yields an $\{R, Q\}|HM|\{C_{\max}, \ell_2, \sum w_j C_j\}$ instance where roughly $p_{\max}^k \cdot m$ jobs are left to be scheduled and which can be solved using Theorem 3.1. To adapt the model (1)–(2) for uniformly related machines, one has a single vector $\mathbf{p} \in \mathbb{N}^\tau$ of "unscaled" processing times, and the right-hand side of constraint (2) becomes $\lfloor T \cdot s_i \rfloor$ for a machine of speed s_i. For ℓ_2, the objective f of the N-fold formulation becomes $f(\mathbf{x}) = \sum_{i=1}^{m} (\mathbf{p}^i \mathbf{x}^i)^2$ which is almost separable convex. (One needs to add an auxiliary variable z^i and a constraint $z^i = \mathbf{p}^i \mathbf{x}^i$ to express it as separable.) For $\sum w_j C_j$, the modification is analogous but slightly more complicated; the approach is identical to the one described by Knop and Koutecký (2018).

It is an open problem whether the $p_{\max}^{\mathcal{O}(k^2)}$ parameter dependence can be improved: Even in the setting with short jobs where $p_{\max} \leq k$, the best algorithm for $Q|HM|C_{\max}$ has a dependence of k^{k^2} (Knop et al., 2019; Knop & Koutecký, 2018).

Proof of Theorem 3.7 It is known (Knop and Koutecký, 2018; Knop et al., 2019) that the problems $\{R, Q\}|HM|\{C_{\max}, \ell_2\}$ have N-fold IP formulations with parameters $N = m$, $\|A\|_\infty = p_{\max}$, $t = k + 1$, $s = 1$, $r = k$. For the $\sum w_j C_j$ objective, $s = p_{\max}$. Applying Proposition 3.4 then shows the claims. We note that it is possible to get a $p_{\max}^{k^2}$ parameter dependence at the cost of a quadratic dependence on N by exploiting the fact that the constraint matrix has small treewidth, see (Knop and Koutecký, 2022, Section 3.3). We are confident that a near-linear algorithm can also be constructed for this case, but this is beyond the scope of this work. □

High multiplicity of machines

So far, for the sake of simplicity, we have only considered high multiplicity of jobs, but not machines. Such a model is also studied: Let κ be the number of *kinds* of machines, where two machines are of the same kind if every job has the same processing time on both machines. The input to a scheduling problem is then represented by a vector $\mathbf{n} \in \mathbb{N}^d$ of job multiplicities, and a vector $\mathbf{m} \in \mathbb{N}^\kappa$ of machine multiplicities.

Theorem 3.7 can be easily extended to this setting by applying the FPT algorithm for *high-multiplicity N-fold IPs* (Knop et al., 2023) instead of Proposition 3.4. A high-multiplicity encoding of an N-fold IP instance is such that we group blocks by τ *types* and only give a description of each type together with a vector of multiplicities $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_\tau)$ of block types; analogously to the definition of machine kinds, two blocks are of the same type if they have an identical right-hand-side vector, lower and upper bounds, block matrices, and objective functions. The algorithm of Knop et al. (2023) runs in time polynomial in τ and $\log \|\boldsymbol{\mu}\|_1$, so applying it to the scheduling problems studied here gives algorithms polynomial in κ and $\log \|\mathbf{m}\|_1$. The parameter dependencies of Knop et al. (2023) are the same as in Proposition 3.4.

4 Hardness

In this section, we present our hardness results, which are all based on a reduction from a tight instance of Balanced Bin Packing to scheduling in the high-multiplicity setting. In this reduction, the number of bins and the number of job types have a linear dependence and for both parameters, the corresponding problems are W[1]-hard. Similarly, the number of items equals the number of machines. For convenience, we hence use k and m as the number of bins and items, respectively, when we describe our hardness instance. If we say, we *reduce* a problem to another problem, we always refer to a polynomial-time reduction.

4.1 Reducing bin packing to balanced bin packing

First, we need to prove that solving Balanced Bin Packing with tight instances remains a hard problem. We do this by reduction from Bin Packing.

Lemma 4.8 Bin Packing *reduces to* Balanced Bin Packing *such that (a)* $a'_{\max} = a_{\max} + 1$, *(b)* $B' = B + n$, *(c)* $k' = k$, *(d)* $n' = nk$, *and (e) tightness is preserved, where* n', k', B', *and* $a'_{\max}$ *are the parameters of the new* Balanced Bin Packing *instance.*

Proof Given an instance of Bin Packing, we obtain an instance of Balanced Bin Packing by increasing the size of each item by 1, setting the new bin capacity to be $B' = B + n$, and adding $n(k-1)$ new items of size 1. Observe that all items of size 1 are "new" items. It is also clear that $a'_{\max} = a_{\max} + 1$.

To show that we preserve feasibility of instances, take any solution of the Bin Packing instance and add new items of size zero such that each bin contains precisely n items. Now, if we increase the size of each item by 1 (including the new items of size zero) and the size of each bin by n, we have obtained a feasible instance of the newly constructed Balanced Bin Packing instance.

For the other direction, consider a solution of such a Balanced Bin Packing instance, subtract 1 from the size of each item and n from the capacity of each bin—note that

there are n items per bin—and remove items of size zero. This is a solution to the instance of Bin Packing.

Regarding tightness, note that the sum of item sizes has increased by exactly nk because we have increased the size by 1 for n "old" items, and added $n(k-1)$ "new" items of size 1. Hence, if the total size of items of the original instance was kB, it became $kB + nk = k(B+n)$, and since $B' = B+n$ is the new bin capacity, the Balanced Bin Packing instance is tight if and only if the Bin Packing instance was tight. □

By Lemma 4.8 and the result by Jansen et al. (2013), we obtain the following corollaries.

Corollary 4.9 Balanced Bin Packing *is NP-hard, even for tight instances.*

Corollary 4.10 Unary Balanced Bin Packing *is* W[1]*-hard parameterized by the number of bins, even for tight instances.*

4.2 Hardness of $Q||C_{\max}$ and $R||C_{\max}$

Let us describe our hardness instance I. Given a tight instance of Balanced Bin Packing with k bins of capacity B and m items, all items sum up to $\sum_{i\in[m]} a_i = k \cdot B =: A$. We construct a $Q|HM|C_{\max}$ instance with m machines and $3k$ job types.

The high-level idea is as follows. We use machine M_i to encode the assignment of item a_i to a bin, so we have m machines. We have job types α_j^1, α_j^0 (we will refer to both of them as $\alpha_j^\times$), and β_j for $j \in [k]$; we refer to a job of type $\alpha_j^\times$ for any j as a job of type α or an α-type job, and similarly for β. For the sake of simplicity, we sometimes do not distinguish between a job and a job type, e.g., by executing $\alpha_j^\times$, we mean executing a job of type $\alpha_j^\times$.

Our goal is to ensure that a specifically assembled schedule, which we call henceforth *perfect*, is optimal. In a perfect schedule, M_i gets, for some $j \in [k]$, precisely a_i times a job of type α_j^1, $A - a_i$ times a job of type α_j^0 and once a job of type β_j. There is no other job on M_i. This corresponds to putting a_i to the j-th bin. Hence, for each $j \in [k]$, there are m/k machines[2] where only jobs of types α_j^1, α_j^0 and β_j appear together and they represent a packing of the corresponding items to the j-th bin.

Let us specify the parameters of I. The target makespan is $T = 3kA^3$; note that we will show that the feasible schedules are precisely the perfect schedules and they have the property that each machine finishes *exactly* at time T. Jobs of type β are by far the largest on all machines. As processing times, we set, for $j \in [k]$,

$$p_{\alpha_j^1} = kA^2 + A(k-j) + 1\,, \qquad p_{\alpha_j^0} = kA^2 + A(k-j)\,,$$
$$p_{\beta_j} = 2kA^3 - A^2(k-j)\,;$$

note that as j increases, so does p_{β_j}. Complementary to p_{β_j}, as j increases, $p_{\alpha_j^\times}$ decreases. To show hardness of $Q||C_{\max}$, we give each machine M_i a specific speed depending on a_i. The *unscaled load of a machine* M_i, denoted $\bar{L}_i$, is the sum of sizes of jobs assigned to M_i before speed scaling. In a perfect schedule, it is

$$\begin{aligned}\bar{L}_i^* &= a_i(kA^2 + A(k-j) + 1) + (A - a_i)(kA^2 + A(k-j)) \\ &\quad + 2kA^3 - A^2(k-j) \\ &= A(kA^2 + A(k-j)) + a_i + 2kA^3 - A^2(k-j) \\ &= 3kA^3 + a_i = T + a_i\,. \end{aligned} \tag{3}$$

The machine speed s_i of machine M_i is

$$s_i = \frac{T + a_i}{T} = \frac{3kA^3 + a_i}{3kA^3}\,.$$

Observe that, in a perfect schedule, each machine M_i finishes exactly by time

$$\frac{\bar{L}_i^*}{s_i} = \frac{T + a_i}{\frac{T+a_i}{T}} = T = 3kA^3\,. \tag{4}$$

The sizes of jobs of type α_j^1 and α_j^0 are almost identical, except jobs of type α_j^1 are slightly longer. For each $j \in [k]$, we have job multiplicities

$$n_{\alpha_j^1} = \frac{A}{k} = B, \qquad n_{\alpha_j^0} = \frac{Am}{k} - B = \frac{(m-1)A}{k},$$
$$n_{\beta_j} = \frac{m}{k}\,.$$

Lemma 4.11 Balanced Bin Packing *with tight instances reduces to* $Q|HM|C_{\max}$ *such that (a) the number of machines equals the number of items, (b) the number of job types equals* $3k$*, where* k *is the number of bins, (c) the job sizes and job multiplicities are bounded by* $\mathcal{O}(A^4)$*, where* A *is the sum of all items of the input instance, (d) the machine speeds are rational numbers with numerator and denominator in* $\mathcal{O}(A^4)$*, and (e) the feasible schedules are precisely the perfect schedules, in which all machines finish exactly at time* $T = 3kA^3$*.*

Proof Clearly, all involved numbers are in $\mathcal{O}(A^4)$ (w.l.o.g. we assume $k, m \in \mathcal{O}(A)$). The other parameters are clear from the description of the hardness instance I above. It remains to prove that the feasible schedules are precisely

[2] Which is an integer by the fact that any Balanced Bin Packing instance must have a number of items divisible by k in order to be feasible.

the perfect schedules (which, in turn, correspond to solutions of the BALANCED BIN PACKING instance). On the one hand, if there is a solution $\mathcal{S}$ of the corresponding instance of BALANCED BIN PACKING, we construct a (feasible) perfect schedule for I as follows. If, in $\mathcal{S}$, a_i is assigned to the j-th bin, we assign to machine M_i, a_i jobs of type α_j^1, $A - a_i$ jobs of type α_j^0, and one job of type β_j. According to Eqs. (3) and (4), this assignment has makespan T and, clearly, all jobs are assigned to some machine.

On the other hand, assume that I is feasible, meaning there is an assignment of jobs to machines not exceeding the target makespan T. Let us analyze the structure of such a schedule σ. First we observe that instead of considering for a machine M_i the makespan T, which is the sum of jobs lengths divided by its speed s_i, we can equivalently consider $T \cdot s_i = T + a_i$ as its capacity—this is the sum of (unscaled) jobs lengths it can process.

Per machine, there is exactly one job of type β_j for some $j \in [k]$, since we can execute at most one β-type job on each machine[3] and we need to place m β-type jobs onto m machines. So each machine is in one set $\mathcal{M}_j$, where $\mathcal{M}_j$ is a set of m/k machines that process a job of type β_j.

Having scheduled a β-type job to a machine, we can execute on this machine at most A α-type jobs.[4] In total, there are Am α-type jobs. Thus, there are exactly A α-type jobs on each machine. Let $j \in [k]$. When we have analyzed the perfect schedules, we have seen how to fit A jobs of type $\alpha_j^\times$ onto a machine from $\mathcal{M}_j$. Observe that, due to the length of the α-type jobs, we can only have a job $\alpha_{j'}^\times$ where $j' < j$ on a machine from $\mathcal{M}_j$ if we also have a job $\alpha_{j''}^\times$ where $j'' > j$. Therefore, there need to be A jobs of type $\alpha_k^\times$ on each machine from $\mathcal{M}_k$. Thus, all jobs of type $\alpha_k^\times$ have to be executed by machines in $\mathcal{M}_k$. Consequently, we need to execute A jobs of type $\alpha_{k-1}^\times$ on each machine of $\mathcal{M}_{k-1}$ since there are no more jobs of type $\alpha_k^\times$ available. This argument inductively propagates for all $j = k, k-1, k-2, \ldots, 1$. Hence, on each machine, the remaining space is at most[5] $a_{\max} < A < p_t$ for any job type t, so no other job can be scheduled. Consider the sizes of the jobs that need to be executed on a machine. For each $i \in [m]$, there can be at most a_i jobs of type α_j^1 on machine M_i. We have, for each $j \in [k]$,

$$\frac{A}{k} \leq \sum_{M_i \in \mathcal{M}_j} a_i \tag{5}$$

because all A/k jobs of type α_j^1 are assigned to machines of $\mathcal{M}_j$. Moreover, we have

$$\sum_{j \in [k]} \sum_{M_i \in \mathcal{M}_j} a_i = A \ .$$

So if there was a $j \in [k]$ with $A/k < \sum_{M_i \in \mathcal{M}_j} a_i$, then there would be a $j' \in [k]$ with $A/k > \sum_{M_i \in \mathcal{M}_{j'}} a_i$. Since this would contradict Eq. (5), we have

$$\sum_{M_i \in \mathcal{M}_j} a_i = \frac{A}{k} = B$$

and a_i jobs of type α_j^1 on each $M_i \in \mathcal{M}_j$ for each $j \in [k]$. Hence, σ is perfect and the sets $\{a_i \mid M_i \in \mathcal{M}_j\}$ for each $j \in [k]$ are a solution for the corresponding instance of BALANCED BIN PACKING. □

It is easy to adjust the hardness instance I of $Q|HM|C_{\max}$ to an instance I_R of $R|HM|C_{\max}$. Instead of machine speeds depending, for machine M_i, on a_i, we will use a larger makespan T_R to host a new "blocker" job type γ, whose length is machine-dependent and leaves space $T + a_i$ on each machine—previously the capacity on a machine with speed s_i. A perfect schedule for I_R is the same as a perfect schedule for I, but with an additional job of type γ assigned once to each machine.

Lemma 4.12 BALANCED BIN PACKING *with tight instances reduces to* $R|HM|C_{\max}$ *such that (a) the number of machines equals the number of items, (b) the number of job types equals* $3k + 1$*, where* k *is the number of bins, (c) the job sizes and job multiplicities are bounded by* $\mathcal{O}(A^4)$*, where* A *is the sum of all items of the* BALANCED BIN PACKING *instance, (d) the feasible schedules are precisely the perfect schedules, in which all machines finish exactly at time* $T_R = 7kA^3$*, and (e) the job sizes matrix* $\mathbf{p}$ *has rank* 2.

Proof In the new hardness instance I_R for $R|HM|C_{\max}$, we use the same job types with the same lengths and multiplicities as in I, which is our hardness instance for $Q|HM|C_{\max}$. We introduce a new job type γ with

$$p_\gamma^i = 4kA^3 - a_i, \qquad n_\gamma = m \ .$$

Observe that γ is the only job type that is machine-dependent. However, its variation between machines is only $-a_i$, which

[3] The smallest β-type job is β_1, which has length $2kA^3 - A^2(k-1)$. Two jobs of type β_1 on the same machine require a capacity of at least $2 \cdot (2kA^3 - A^2(k-1)) = 4kA^3 - 2A^2(k-1) > 3kA^3 + kA^3 - 2kA^2 = 3kA^3 + kA^2(A-2k)$, which is still larger than the capacity $3kA^3 + a_i$ of any machine M_i as $A > 2k$ (otherwise the BALANCED BIN PACKING instance is trivial) and $kA^2 > a_{\max}$.

[4] The smallest β-type job is β_1, which has length $2kA^3 - A^2(k-1)$. The smallest α-type job is α_k^0, which has length kA^2. One job of type β_1 and $A+1$ jobs of type α_k^0 on the same machine require a capacity of at least $2kA^3 - A^2(k-1) + (A+1)kA^2 = 3kA^3 + A^2$, which is still larger than the capacity $3kA^3 + a_i$ of any machine M_i as $A^2 > a_{\max}$.

[5] This maximum can only be reached if there are A jobs of type α_j^0 and no jobs of type α_j^1 on a machine.

is relatively small compared to its total length. Again, the parameters are clear from the definition of I_R and we prove the correctness of (d) and (e) next.

On the one hand, if there is a solution $\mathcal{S}$ of the corresponding instance of Balanced Bin Packing, we construct a perfect scheduling for I as follows. If, in $\mathcal{S}$, a_i is assigned to the j-th bin, we assign, to machine M_i, a_i jobs of type α_j^1, $A - a_i$ jobs of type α_j^0, one job of type β_j, and one job of type γ. This assignment has makespan T_R and, clearly, all jobs are assigned to some machine.

On the other hand, assume that I_R is feasible, meaning there is a schedule σ not exceeding the target makespan T_R. Again, let us analyze the structure of such a solution. Per machine, there is exactly one job of type γ since we can execute at most one such job on each machine. The space remaining on machine M_i after executing a job of type γ is

$$T_R - p^i_\gamma = 7kA^3 - (4kA^3 - a_i) = 3kA^3 + a_i \ .$$

This is precisely the capacity of machine M_i in I as described in the proof of Lemma 4.11. After scheduling all jobs of type γ, there are also the same job types with the same lengths and multiplicities remaining. Thus, the rest of the analysis is the same.

It remains to show that the rank of the job sizes matrix $\mathbf{p}$ is 2. Define a matrix C whose rows are indexed by the job types as follows. The row for job type $t \in \{\alpha_j^0, \alpha_j^1, \beta_j\}$ (for every $j \in [k]$) is $(p_t, 0)$, and the row for γ is $(4kA^3, -1)$. Next, define a matrix D whose columns are indexed by the machines as follows: column $i \in [m]$ is $(1, a_i)$. It is easy to verify that $C \cdot D = \mathbf{p}$. □

Applying the reductions of Lemmas 4.11 and 4.12 to Balanced Bin Packing with 2 bins, we have that $Q|HM|C_{\max}$ and $R|HM|C_{\max}$ are NP-hard with 6 and 7 job types, respectively. $R|HM|C_{\max}$ can be reduced to 4 job types, and similar ideas can be used to improve the previously described reduction to only require $3k - 2$ job types.

Theorem 4.13 *$Q|HM|C_{\max}$ is NP-hard already with 6 job types.*

Theorem 4.14 *$R|HM|C_{\max}$ is NP-hard already with 4 job types and with $\mathbf{p}$ of rank 2.*

Proof We will modify the reduction described in Lemma 4.12 to use only four job types if the number of bins $k = 2$. First, we remove the job type γ to get to six different job types. Recall that $p^i_\gamma = 4kA^3 - a_i$. For the $4kA^3$, we will account for when adjusting the makespan and we add the $-a_i$ to the β-type jobs (now $p^i_{\beta_j} = 2kA^3 - A^2(k - j) - a_i$). Second, we blow up the makespan by a factor of $A/(7k)$. So we have $T = A^4$.

To reduce to five different job types, we remove all jobs of type β_1. Still, we want A times a job of type $\alpha_1^\times$ on every machine of $\mathcal{M}_1$. So its size will be around A^3. To distinguish between α_1^1 and α_1^0 and get a dependency of machine M_i on item a_i, we add $A - a_i$ and subtract a_i, respectively. So, for $i \in [m]$, we have

$$p^i_{\alpha_1^1} = A^3 + A - a_i \qquad \text{and} \qquad p^i_{\alpha_1^0} = A^3 - a_i \ . \quad (6)$$

As in the previous reduction, we can fit a_i times $p^i_{\alpha_1^1}$ and $A - a_i$ times $p^i_{\alpha_1^0}$ to machine M_i, which then needs precisely the makespan T since $a_i \cdot (A^3 + A - a_i) + (A - a_i) \cdot (A^3 - a_i) = A^3 a_i + A a_i - a_i^2 + A^4 - A a_i - A^3 a_i + a_i^2 = A^4 = T$.

To reduce to four different job types, we remove all jobs of type α_2^0 and we change the length of α_2^1 to

$$p^i_{\alpha_2^1} = A^2 \quad (7)$$

for all $i \in [m]$. We lengthen the job of type β_2 to

$$p^i_{\beta_2} = A^4 - a_i A^2 \ , \quad (8)$$

which is the makespan T minus a_i times $p^i_{\alpha_2^1}$. Note that the rank of $\mathbf{p}$ is still just 2: The rows of C are $(A^3 + A, -1)$ for α_1^1, $(A^3, -1)$ for α_1^0, $(A^2, 0)$ for α_2^1, and $(A^4, -A^2)$ for β_2, and D is defined as before.

It remains to show the correctness of this reduction. Clearly, if there is a solution to the instance of Balanced Bin Packing with two bins (i.e., a partition), we can assign the jobs to the machines as in the perfect schedule from Lemma 4.11 ignoring β_1 and α_2^0.

Assume that there is a solution of the obtained instance of $R|HM|C_{\max}$. On half of the machines, there is a job of type β_2. On these machines, namely $\mathcal{M}_2$, there is no space for a job of type $\alpha_1^\times$. So, all $Am/2$ jobs of type $\alpha_1^\times$ are scheduled to the $m/2$ machines of $\mathcal{M}_1$. As there cannot be more than A jobs of type $\alpha_1^\times$ on a machine, there are precisely A jobs of type $\alpha_1^\times$ on each machine M_i of $\mathcal{M}_1$—at most a_i of them can be α_1^1. To place all $A/2$ jobs of type α_1^1, $\mathcal{M}_1$ needs to be chosen such that the corresponding item sizes in the Balanced Bin Packing instance sum up to at least $A/2$. The free space on a machine from $\mathcal{M}_1$ is at most $a_{\max} A < A^2$, so there is no job of type α_2^1.

To schedule all $A/2$ jobs of type α_2^1, $\mathcal{M}_2$ needs to be chosen such that the corresponding item sizes in the Balanced Bin Packing instance sum up to at least $A/2$. As the total sum of items is A, both partitions correspond to items summing up to precisely $A/2$. This yields an equal partition of the items. □

The complexity of $Q|HM|C_{\max}$ ($R|HM|C_{\max}$) with less than 6 (4) job types remains open. From Lemmas 4.11 and 4.12 and Corollary 4.10, we get our main result:

Theorem 4.15 *$X||C_{\max}$ is* W[1]*-hard parameterized by the number of job types with (a) $X = Q$ and even when* $\mathbf{n}$, $\mathbf{p}$, *and* $\mathbf{s}$ *are given in unary; (b) $X = R$ and even when* $\mathbf{n}$ *and* $\mathbf{p}$ *are given in unary and* $\operatorname{rank}(\mathbf{p}) = 2$.

4.3 NP-hardness of Cutting Stock

We can transfer our hardness result for $Q|HM|C_{\max}$ to the related problem Cutting Stock.

Cutting Stock

Input: k item types of sizes $\mathbf{p} = (p_1, \ldots, p_k) \in \mathbb{N}^k$ and multiplicities $\mathbf{n} = (n_1, \ldots, n_k) \in \mathbb{N}^k$, m bin types with sizes $\mathbf{s} = (s_1, \ldots, s_m) \in \mathbb{N}^m$ and costs $\mathbf{c} = (c_1, \ldots, c_m) \in \mathbb{N}^m$.

Find: A vector $\mathbf{x} = (x_1, \ldots, x_m) \in \mathbb{N}^m$ of how many bins to buy of each size, and a packing of items to those bins, such that the total cost $\mathbf{cx}$ is minimized.

As it seems natural, we associate the job types, job multiplicities, and machines with the item types, item multiplicities, and bin types, respectively. The difficulty lies in enforcing that each bin type is used exactly once.

Lemma 4.16 *$Q|HM|C_{\max}$ with k job types and m machines reduces to* Cutting Stock *with $k + 2$ item types and m bin types.*

Proof We will set the sizes of bin types as 3-dimensional vectors, whose interpretation as numbers is straightforward by choosing the base of each coordinate sufficiently large to prevent carry when summing. For machine M_i with capacity $T + a_i$, we add a bin type of size and cost $(1, 2^{i-1}, T + a_i)$. For each original job type t of size p_t, there is an item type of size $(0, 0, p_t)$ with the same multiplicity n_t. We will add two new item types: There are m items of type η which have size $(1, 0, 0)$, and $2^m - 1$ items of type ν which have size $(0, 1, 0)$. The target cost is $C = (m, 2^m - 1, mT + A)$.

Clearly, a feasible schedule translates easily to a packing: Buying each bin type exactly once costs exactly C, the original item types are packed according to the feasible schedule, and we pack one η-type job and 2^{i-1} ν-type jobs on machine M_i.

In the other direction, first notice that we have to use at least m bins to pack the η-type jobs, and at most m bins are affordable due to the budget C. We want to show that we have to use each bin type exactly once. Focus on the second coordinates of the 3-dimensional vectors. Since the total size of items with respect to these coordinates is $2^m - 1$, which is precisely the affordable capacity, a solution to Cutting Stock must buy m bins with capacity $2^m - 1$. This is equivalent to decomposing the number $2^m - 1$ into a sum of some m numbers which are powers of 2, namely $2^0, 2^1, \ldots, 2^{m-1}$. Clearly, the unique decomposition is $2^m - 1 = 2^0 + 2^1 + \cdots + 2^{m-1}$. Hence, the unique way to obtain capacity C by buying m bins is to buy one bin of each type, concluding the proof. □

Note that the W[1]-hardness of $Q||C_{\max}$ does not immediately imply W[1]-hardness of Cutting Stock when $\mathbf{p}$, $\mathbf{n}$, $\mathbf{c}$ are given in unary, because the construction of Lemma 4.16 blows up each of $\mathbf{p}$, $\mathbf{n}$, $\mathbf{c}$: It introduces large costs, items η with large size, and items ν with large multiplicity. Using our hardness of $Q|HM|C_{\max}$ with 6 job types together with Lemma 4.16 yields:

Theorem 4.17 Cutting Stock *is NP-hard already with* 8 *item types.*

4.4 Hardness of $Q||\ell_2$ and $R||\ell_2$

We will now transfer our hardness reduction to the ℓ_2 norm. Remember that, in our hardness instance I, the speed s_i of machine M_i depends linearly on $T + a_i$ (normalized by $1/T$ for all machines). For the ℓ_2 norm, we observe that the machine speed affects the objective value by its square. So for a machine where we double its speed, it contributes only a fourth to the objective value. Then, one can construct an instance where it is more beneficial to schedule more than the loads of a perfect schedule to the faster machines leaving the slower machines rather empty.

To still apply our argument that the perfect schedules, which precisely correspond to bin packings, are the only ones admitting an optimal schedule, we adjust the machine speeds. It should be a value in the order of $\sqrt{T + a_i}$. We use the ceiling function to have rational machine speeds. However, for our reduction, it is crucial that machines M_i and M_j have different speeds if $a_i \neq a_j$. To make each $\lceil \sqrt{T + a_i} \rceil$ different from $\lceil \sqrt{T + a_i + 1} \rceil$, we scale up $\sqrt{T + a_i}$ by a sufficiently large factor. We will see that we can set this factor to be $(T + a_{\max})$, which results, for machine M_i, in a new machine speed of

$$s_i = \left\lceil (T + a_{\max})\sqrt{T + a_i} \right\rceil . \tag{9}$$

In the following, we will use ℓ_2^2, which is the square of the ℓ_2 norm and is isotonic to it. Recall that the unscaled load of M_i is $\bar{L}_i = L_i \cdot s_i = \sum_{t=1}^{\tau} p_t^i x_t^i$, where $\mathbf{x}^i = (x_1^i, \ldots, x_\tau^i)$ is the vector of job multiplicities scheduled to machine M_i, and τ is the number of job types.

Lemma 4.18 Balanced Bin Packing *with tight instances reduces to $Q|HM|\ell_2^2$ with the properties of Lemma 4.11 via the hardness instance I with modified $s_i = \lceil (T + a_{\max})\sqrt{T + a_i} \rceil$ and target value $\sum_{i=1}^{m} ((T + a_i)/s_i)^2$.*

Proof As before, if the instance of BALANCED BIN PACKING has a solution where item a_i is assigned to the j-th bin, we construct a perfect schedule, where we assign a_i jobs of type α_j^1, $A - a_i$ jobs of type α_j^0 and one job of type β_j to machine M_i. As this gives us load $(T + a_i)/s_i$ on machine M_i, we reach precisely the target objective value $\sum_{i=1}^{m} ((T + a_i)/s_i)^2$ for the ℓ_2^2 objective.

For the other direction, assume there is a schedule σ of jobs to machines such that the objective value is at most $\sum_{i=1}^{m} ((T + a_i)/s_i)^2$. We distinguish two cases.

Case 1: The unscaled load of machine M_i is $T + a_i$, for each $i \in [m]$. Observe that the objective value of σ equals the prescribed threshold objective value $\sum_{i=1}^{m} ((T + a_i)/s_i)^2$. By Lemma 4.11 (e), we know that such a schedule is perfect and exists if and only if there is a solution to the corresponding BALANCED BIN PACKING instance.

Case 2: There is an $i \in [m]$ such that M_i has unscaled load different from $T + a_i$. Consider the unscaled loads $\mathcal{L} = (\bar{L}_1, \ldots, \bar{L}_m)$ scheduled to each of the machines in σ. Since the total unscaled load is independent of the schedule, we can reach $\mathcal{L}$ from the "perfect" unscaled load distribution $(T + a_1, \ldots, T + a_m)$, which a perfect schedule (as it appears in Case 1) would admit, by iteratively moving a portion of the load from one machine to another. Note that we do not speak of moving jobs here. For this argument, we only consider the unscaled load of each machine as an integral number and ignore the jobs. In this process …

- m iterations of re-distribution are sufficient; in each step we take the machine M_i with the smallest deviation Δ_i from $\bar{L}_i$ (so, initially $\Delta_i = |(T + a_i) - \bar{L}_i|$) and move Δ_i integral units of load from or to M_i depending on the direction of the deviation. Note that there exists some other machine M_j to/from which to move because we chose i to minimize Δ_i.
- the load of each machine monotonously increases, decreases, or remains unchanged, i.e., we do not first add and then remove a portion of load or the other way around.

We show that in every step the objective value only increases, and hence, this case cannot occur as we already matched the threshold objective value in the "perfect" distribution of Case 1.

Consider one such step. We move load $r \geq 1$ to machine M_i and take it from machine M_j. Before, we have already moved in total $z_i \geq 0$ to M_i and we have already removed in total $z_j \geq 0$ from M_j. If M_i is slower than M_j, then the objective value definitely increases. Hence, we assume $s_i \geq s_j$ (this implies $a_i \geq a_j$). So it remains to show that

$$
\begin{aligned}
&\frac{(T + a_i + z_i)^2}{s_i^2} + \frac{(T + a_j - z_j)^2}{s_j^2} < \frac{(T + a_i + z_i + r)^2}{s_i^2} \\
&\quad + \frac{(T + a_j - z_j - r)^2}{s_j^2} \\
&\Leftrightarrow \frac{(T + a_j - z_j)^2}{s_j^2} - \frac{(T + a_j - z_j - r)^2}{s_j^2} \\
&\quad < \frac{(T + a_i + z_i + r)^2}{s_i^2} - \frac{(T + a_i + z_i)^2}{s_i^2} \\
&\Leftrightarrow s_i^2 \left(2r(T + a_j - z_j) - r^2\right) \\
&\quad < s_j^2 \left(2r(T + a_i + z_i) + r^2\right) \\
&\Leftrightarrow \frac{s_i^2}{s_j^2} < \frac{2r(T + a_i + z_i) + r^2}{2r(T + a_j - z_j) - r^2} \\
&\Leftrightarrow \frac{s_i^2}{s_j^2} < \frac{2(T + a_i + z_i) + r}{2(T + a_j - z_j) - r} \,. \qquad (10)
\end{aligned}
$$

Next, we analyze the machine speed s_i as defined in Eq. (9). Recall that we scale up $\sqrt{T + a_i}$ by a sufficiently large factor b to make each $\lceil \sqrt{T + a_i} \rceil$ different from $\lceil \sqrt{T + a_i + 1} \rceil$. If the difference between $\sqrt{T + a_i}$ and $\sqrt{T + a_i + 1}$ is at least d, then it must hold that

$$
b > \frac{1}{d} \geq \frac{1}{\sqrt{T + a_{\max} + 1} - \sqrt{T + a_{\max}}} \,.
$$

We have chosen $b = T + a_{\max}$, since $x > 1/(\sqrt{x + 1} - \sqrt{x})$ for $x \geq 5$. Hence, we conclude

$$
\lceil (T + a_{\max}) \sqrt{T + a_i} \rceil < (T + a_{\max}) \sqrt{T + a_i + 1} \,. \qquad (11)
$$

Furthermore, note that for three positive numbers w, x, and y with $x \geq y > w$, it always holds that

$$
\begin{aligned}
\frac{x + 2w}{y} &= \frac{\frac{y-w}{y}(x + 2w)}{y - w} = \frac{(1 - \frac{w}{y})(x + 2w)}{y - w} \\
&= \frac{x - \frac{wx}{y} + 2w - \frac{2w^2}{y}}{y - w} \\
&< \frac{x + w(2 - 1 - 0)}{y - w} = \frac{x + w}{y - w} \,. \qquad (12)
\end{aligned}
$$

With these inequalities in hand, we finally show the correctness of inequality (10):

$$
\begin{aligned}
\frac{s_i^2}{s_j^2} &\overset{(9)}{=} \frac{\lceil (T + a_{\max}) \sqrt{T + a_i} \rceil^2}{\lceil (T + a_{\max}) \sqrt{T + a_j} \rceil^2} \\
&\overset{(11)}{<} \frac{\left((T + a_{\max}) \sqrt{T + a_i + 1}\right)^2}{\left((T + a_{\max}) \sqrt{T + a_j + 1}\right)^2}
\end{aligned}
$$

$$= \frac{T + a_i + 1}{T + a_j + 1} < \frac{T + a_i + 1}{T + a_j} = \frac{2(T + a_i) + 2}{2(T + a_j)}$$
$$\overset{(r \geq 1)}{\leq} \frac{2(T + a_i) + 2r}{2(T + a_j)}$$
$$\overset{(12),(a_i \geq a_j),(T > r)}{<} \frac{2(T + a_i) + r}{2(T + a_j) - r}$$
$$\leq \frac{2(T + a_i + z_i) + r}{2(T + a_j - z_j) - r}$$

□

Similarly, we can transfer our hardness instance to $R|HM|\ell_2^2$.

Lemma 4.19 BALANCED BIN PACKING *with tight instances reduces to* $R|HM|\ell_2^2$ *with the properties of Lemma 4.12 via the hardness instance* I_R *with target value* $m \cdot T_R^2$.

Proof Again, if the instance of BALANCED BIN PACKING has a solution where item a_i is assigned to the j-th bin, we construct a perfect schedule, where we schedule a_i jobs of type α_j^1, $A - a_i$ jobs of type α_j^0, one job of type β_j, and one job of type γ to machine M_i. As this gives us processing time T_R per machine, we precisely reach the target objective value of mT_R^2 for the ℓ_2^2 objective.

For the other direction, assume there is a schedule of jobs to machines such that the objective value is at most $mT_R^2 = 49\,mk^2A^6$. We distinguish three cases.

Case 1: *The load of each machine is at most* $T_R = 7kA^3$. Such a schedule would thus have makespan T_R and is feasible for $R|HM|C_{\max}$ with target makespan T_R. By Lemma 4.12, we know that such a schedule exists if and only if there is a solution to the corresponding BALANCED BIN PACKING instance. By property (d) of Lemma 4.12, it admits an objective value of precisely mT_R^2 for the ℓ_2^2 objective.

Case 2a: *There is a machine with load* $T_R' > T_R = 7kA^3$, *and on each machine there is precisely one job of type* γ. Since the processing time for all α- and β-type jobs is the same on all machines and we have exactly one job of type γ per machine, the total load is independent of the schedule and is $m \cdot T_R$. Fixing the total load, the ℓ_2^2 objective reaches its minimum uniquely by distributing the load evenly; see, e.g., Knop and Koutecký (2018, Proof of Theorem 3). Thus, the objective mT_R^2 can only be reached if the load of every machine is T_R, so this case cannot occur.

Case 2b: *There is a machine which schedules at least two jobs of type* γ. In this case, we exploit Claim 4.20, which we prove next. Again, it contradicts our assumption of σ having objective value at most mT_R^2. So this case can also not occur. □

Claim 4.20 *Any schedule in Case 2b has objective value strictly greater than* $r \cdot mT_R^2$ *with* $r = (m - 0.98)/(m - 1)$. *Hence, the objective value of such a schedule exceeds* mT_R^2 *by at least*

$$(r-1)mT_R^2 = \frac{0.02}{m-1} \cdot 49mk^2A^6 > 0.98k^2A^6 \ .$$

Proof The dependence of p_γ^i on the choice of a machine M_i is only subtracting a_i. So we get a lower bound on the total sum of job sizes of all jobs in any schedule if we subtract m times $a_{\max}$ (as we have m jobs of type γ). This yields a total sum

$$\begin{aligned}\mathcal{T} = &\sum_{j \in [k]} \left(n_{\alpha_j^1} p_{\alpha_j^1} + n_{\alpha_j^0} p_{\alpha_j^0} + n_{\beta_j} p_{\beta_j} \right) \\ &+ m \left(4kA^3 - a_{\max} \right) \\ =&7mkA^2 + A - ma_{\max} \ .\end{aligned}$$

The machine where we have scheduled two jobs of type γ has load at least

$$T_R' \geq 2 \cdot \left(4kA^3 - a_{\max} \right) > 8kA^3 - 2A \ .$$

This is already greater than T_R, which is in turn at least $\mathcal{T}/m$. Hence, we assume for the rest of the proof that T_R' is exactly $8kA^3 - 2A$ and the remaining processing time $\mathcal{T} - T_R'$ is distributed equally across the other $m - 1$ machines as otherwise the resulting objective value would only increase. The average load L_{avg} of the remaining machines is

$$\begin{aligned}L_{\text{avg}} &= \frac{\mathcal{T} - T_R'}{m-1} = \frac{\left(7mkA^3 + A - ma_{\max}\right) - \left(8kA^3 - 2A\right)}{m-1} \\ &> \frac{7mkA^3 - 8kA^3 - mA}{m-1} \ .\end{aligned}$$

Hence, the objective value of such a schedule is at least

$$\begin{aligned}&\left(8kA^3 - 2A\right)^2 + (m-1)\left(\frac{7mkA^3 - 8kA^3 - mA}{m-1}\right)^2 \\ &> \frac{64mk^2A^6 - 64k^2A^6 - 32mkA^4 + 49m^2k^2A^6 - 112mk^2A^6 - 14m^2kA^4}{m-1} \\ &> 49mk^2A^6 \frac{m - \frac{48}{49} - \frac{64}{m} - \frac{46m}{A}}{m-1} \geq rmT_R^2 \ , \\ &\text{where } \frac{m - \frac{48}{49} - \frac{64}{m} - \frac{46m}{A}}{m-1} \geq r = \frac{m - \frac{49}{50}}{m-1}\end{aligned}$$

because, without loss of generality, we can assume that $m \geq 64 \cdot 4900$ as otherwise we could add more dummy items to our BALANCED BIN PACKING as described in the proof of Lemma 4.8, and we can assume that $A \geq 46 \cdot 4900\,m$ as otherwise we could scale up the items of the BALANCED BIN PACKING instance by a factor of $46 \cdot 4900$. □

The following corollaries follow immediately from Lemmas 4.18 and 4.19; as before, it is likely that one might improve this to 4 job types.

Corollary 4.21 *$X|HM|\ell_2$ is NP-hard already for t job types with (a) $X = Q$, $t = 6$. (b) $X = R$, $t = 7$, and $\mathrm{rank}(\mathbf{p}) = 2$.*

Corollary 4.22 *$X||\ell_2$ is W[1]-hard parameterized by the number of job types with (a) $X = Q$ and even when $\mathbf{n}$, $\mathbf{p}$, and $\mathbf{s}$ are given in unary; (b) $X = R$ and even when $\mathbf{n}$ and $\mathbf{p}$ are given in unary and $\mathrm{rank}(\mathbf{p}) = 2$.*

4.5 Hardness of $R||\sum w_jC_j$

We will define weights in the hardness instance I_R from Lemma 4.12 to also cover $R|HM|\sum w_jC_j$. Denote $\rho_j^i = w_j/p_j^i$ the *Smith ratio* of a job j on machine M_i, where w_j is its weight. It is known that given an assignment of jobs to machines, an optimal schedule is obtained by executing jobs ordered by their Smith ratios (on each machine) non-increasingly (Smith, 1956). It suffices to restrict ourselves to such schedules, and an assignment of jobs to machines describes such a schedule.

We would like to use the same approach as for ℓ_2 (Lemma 4.19) because it is known that $\sum w_jC_j$ and ℓ_2 are often (but not always) closely related. However, because the size of a job of type γ depends on the machine (while its weight does not), it is impossible to express an exact objective value of a perfect schedule from the previous sections. This would make the argument of an analogue of Case 2a of Lemma 4.19 invalid and a no-instance of Balanced Bin Packing might reduce to a yes-instance of $R||\sum w_jC_j$. On the positive side, the contribution of all α- and β-type jobs to the sum of weighted completion times is always the same as they and their weights are machine-independent. If we schedule to each machine exactly one job of type γ, then we will have each machine-dependent processing time once and, across all machines, their contribution is independent of the schedule and we can specify an exact target objective value. Consequently, we can apply the same argumentation for Case 1 and Case 2a as in Lemma 4.19. For Case 2b, we will exploit Claim 4.20 once again and combine it with a gap argument (Lemma 4.23).

To obtain the *weighted hardness instance* I_R^w, we define the following weights for our hardness instance I_R from Sect. 4.2. For each α- and β-type job, the weight equals its processing time and for a job of type γ, it is slightly greater:

$$w_{\alpha_j^\times} = p_{\alpha_j^\times} \qquad w_{\beta_j} = p_{\beta_j}$$
$$w_\gamma = 4kA^3 \ \ (= p_\gamma^i + a_i \text{ for each } i \in [m])$$

Lemma 4.23 *Let σ be any schedule of the weighted hardness instance, let $(L_1, L_2, \ldots, L_m)$ be its load vector, and let $\mathcal{L} = \frac{1}{2}\sum_{i=1}^m {L_i}^2$. Further, let $\Gamma = \frac{1}{2k}\sum_{j=1}^k (Aw_{\alpha_j^1}^2 + (m-1)Aw_{\alpha_j^0}^2 + mw_{\beta_j}^2)$, $\Delta_{\text{linear}}^{1:1} = \frac{1}{2}\sum_{i=1}^m p_\gamma^i w_\gamma$, $\Delta_{\text{quadr}}^{1:1} = \frac{1}{2}\sum_{i=1}^m p_\gamma^i \cdot a_i$, $\Delta^{1:1} = \Delta_{\text{linear}}^{1:1} + \Delta_{\text{quadr}}^{1:1}$, $\Delta_{\text{linear}}^{\min} = \frac{1}{2}m(w_\gamma - a_{\max})w_\gamma$, and $\Delta_{\text{quadr}}^{\min} = \frac{1}{2}m(w_\gamma - a_{\min})a_{\min}$ where $a_{\min} = \min_{i\in[m]} a_i$. Then, the following statements hold:*

(i) The objective value of σ under $\sum w_jC_j$ is at least $\mathcal{L} + \Gamma + \Delta_{\text{linear}}^{\min} + \Delta_{\text{quadr}}^{\min}$.
(ii) If σ schedules one job of type γ per machine, then the objective value of σ under $\sum w_jC_j$ is $\mathcal{L} + \Gamma + \Delta^{1:1}$.

Proof First notice that the Smith ratio of all α- and β-type jobs is 1, and the Smith ratio of the jobs of type γ is strictly greater than 1, so the jobs of type γ will always be executed first. We use the following description of the objective function due to Knop and Koutecký (2018). Assume that τ job types are ordered according to their Smith ratios with respect to some machine M_i (with $i \in [m]$) and let these job types be indexed by $t = 1, \ldots, \tau$. Let x_t^i be the number of jobs of the type indexed by t that are scheduled on machine M_i, and let $z_t^i = \sum_{\ell=1}^t p_\ell^i x_\ell^i$ be the time spent to process the first t job types. Define $\rho_{\tau+1}^i = 0$. Then, the contribution of machine M_i to the total $\sum w_jC_j$ objective is

$$\frac{1}{2}\sum_{t=1}^{\tau}\left[\left(z_t^i\right)^2\left(\rho_t^i - \rho_{t+1}^i\right) + p_t^i w_t x_t^i\right] .$$

In our case, the coefficients of $(z_t^i)^2$ for any α- and β-type job except the last one are 0 because their Smith ratios are identical, hence $\rho_t^i - \rho_{t+1}^i = 0$. The term of the last α- or β-type job has $z_t^i = L_i$ as the load of machine M_i and its coefficient is $\rho_\tau^i - \rho_{\tau+1}^i = 1 - 0$, so this term is $\frac{1}{2}L_i^2$. Hence, summing up those terms over all machines gives $\mathcal{L}$, and we are left to account for (a) the quadratic terms corresponding to the jobs of type γ, and (b) the linear terms $p_t^i w_t x_t^i$.

First, we consider the linear terms, but only for the α- and β-type jobs. Since the sizes of these jobs are independent of the machines, we just sum them up without knowing to which machines they are scheduled. For each $j \in [k]$, we have A/k jobs of type α_j^1, $(m-1)A/k$ jobs of type α_j^0 and m/k jobs of type β_j. Hence, across all $j \in [k]$, this is

$$\frac{1}{2}\sum_{j=1}^{k}\left(\frac{A}{k}\cdot p_{\alpha_j^1}w_{\alpha_j^1} + \frac{(m-1)A}{k}\cdot p_{\alpha_j^0}w_{\alpha_j^0} + \frac{m}{k}\cdot p_{\beta_j}w_{\beta_j}\right)$$
$$= \frac{1}{2k}\sum_{j=1}^{k}\left(Aw_{\alpha_j^1}^2 + (m-1)Aw_{\alpha_j^0}^2 + mw_{\beta_j}^2\right) = \Gamma .$$

Now, we consider the linear and quadratic terms of the jobs of type γ. Let us first assume that we have scheduled exactly

one job of type γ per machine. This means that across all machines, every possible quadratic and linear term appears precisely once. So for the linear terms, we get

$$\frac{1}{2}\sum_{i=1}^{m} p_\gamma^i w_\gamma = \Delta_{\text{linear}}^{1:1} .$$

For the quadratic terms, we get

$$\frac{1}{2}\sum_{i=1}^{m}\left(p_\gamma^i\right)^2\left(\frac{w_\gamma}{p_\gamma^i}-1\right) = \frac{1}{2}\sum_{i=1}^{m} p_\gamma^i\left(w_\gamma - p_\gamma^i\right)$$
$$= \frac{1}{2}\sum_{i=1}^{m} p_\gamma^i \cdot a_i = \Delta_{\text{quadr}}^{1:1} .$$

This shows the correctness of property (ii).

To show property (i), let us now drop the assumption that we have scheduled exactly one job of type γ per machine and determine a lower bound for the objective value of an arbitrary schedule. Still, $\mathcal{L}$ and Γ have the structure described above. Thus, we specify a lower bound by minimizing the linear and the quadratic terms for the jobs of type γ. Clearly, the linear terms are minimum if we schedule all of the m jobs to the machine where it has the smallest size—this is machine M_i corresponding to item $a_{\max}$. Then, since $p_\gamma^i = w_\gamma - a_{\max}$ here, we have

$$\frac{1}{2}m\left(w_\gamma - a_{\max}\right) w_\gamma = \Delta_{\text{linear}}^{\min} .$$

For the quadratic terms, assume that, for each $i \in [m]$, we schedule x_γ^i jobs of type γ to machine M_i. Consequently, the quadratic term of the γ-type jobs over all machines is

$$\frac{1}{2}\sum_{i=1}^{m}\left(x_\gamma^i p_\gamma^i\right)^2\left(\frac{w_\gamma}{p_\gamma^i}-1\right) \overset{(p_\gamma^i = w_\gamma - a_i)}{=} \frac{1}{2}\sum_{i=1}^{m}\left(x_\gamma^i\right)^2$$
$$\left(w_\gamma\left(w_\gamma - a_i\right) - \left(w_\gamma - a_i\right)^2\right)$$
$$= \frac{1}{2}\sum_{i=1}^{m}\left(x_\gamma^i\right)^2\left(w_\gamma^2 - w_\gamma a_i - w_\gamma^2 + 2w_\gamma a_i - a_i^2\right)$$
$$= \frac{1}{2}\sum_{i=1}^{m}\left(x_\gamma^i\right)^2\left(w_\gamma - a_i\right) a_i .$$

Since $w_\gamma \gg a_i$, it is better to schedule more jobs of type γ to the machines representing small a_i and, in particular, $a_{\min}$. On the other hand, scheduling too many jobs to one machine can increase the objective value disproportionately due to $(x_\gamma^i)^2$. The optimal solution might be a trade-off, for which we find the following lower bound (recall $\sum_{i=1}^{m} x_\gamma^i = m$):

$$\frac{1}{2}\sum_{i=1}^{m}\left(x_\gamma^i\right)^2\left(w_\gamma - a_i\right) a_i \geq \frac{1}{2}m\left(w_\gamma - a_{\min}\right) a_{\min} = \Delta_{\text{quadr}}^{\min}$$

□

With this lemma at hand, it is not difficult to show that the weighted hardness instance indeed reduces Balanced Bin Packing to $R|HM|\sum w_j C_j$ as before:

Lemma 4.24 Balanced Bin Packing *with tight instances reduces to* $R|HM|\sum w_j C_j$ *with the properties of Lemma 4.12 via the weighted hardness instance* I_R^w *with target value* $\frac{1}{2}mT_R^2 + \Gamma + \Delta^{1:1}$.

Proof Note that a perfect schedule satisfies the condition that every machine executes exactly one job of type γ and the load of every machine is T_R. Hence by Lemma 4.23, the value of a perfect schedule is precisely the target value.

Now, assume that a schedule σ is given whose $\sum w_j C_j$ objective is at most the target objective. We again distinguish three cases:

Case 1: *The load of each machine is at most* T_R.
This is again a schedule of makespan at most T_R, and the analysis of Lemma 4.12 applies. Hence, σ is a perfect schedule.

Case 2a: *Each machine contains exactly one* γ*-type job, and there is a machine with a load greater than* T_R.
By Lemma 4.23(ii), we know that such a schedule has an objective value of $\mathcal{L} + \Gamma + \Delta^{1:1}$. In this sum, Γ and $\Delta^{1:1}$ are constant and independent of the loads of the machines. We use the same argument as in Lemma 4.19. As the objective value of $\frac{1}{2}mT_R^2 + \Gamma + \Delta^{1:1}$ is matched precisely if the total load is distributed evenly (i.e., Case 1), re-distributing the same total load unevenly increases the quadratic term $\mathcal{L}$. Hence, this case cannot occur.

Case 2b: *There is a machine which schedules at least* 2 *jobs of type* γ.
By Lemma 4.23(i), the objective value of σ is at least $\mathcal{L} + \Gamma + \Delta_{\text{linear}}^{\min} + \Delta_{\text{quadr}}^{\min}$. Let us compare this to the target value $\frac{1}{2}mT_R^2 + \Gamma + \Delta_{\text{linear}}^{1:1} + \Delta_{\text{quadr}}^{1:1}$ summand by summand. As shown in Claim 4.20, we have $(\sum_{i=1}^{m} L_i^2) - mT_R^2 \geq (r-1)mT_R^2$. However, we now have $\mathcal{L} = \frac{1}{2}\sum_{i=1}^{m} {L_i}^2$. For comparing the first summands, we put these things together:

$$\mathcal{L} - \frac{1}{2}mT_R^2 \geq \frac{1}{2}(r-1)mT_R^2 > 0.49k^2A^6$$

Then, of course, Γ is the same in both sums. Consider each of the m summands of $\Delta_{\text{linear}}^{1:1}$. Compared to its counterpart in $\Delta_{\text{linear}}^{\min}$, it is greater by at most $w_\gamma a_{\max} < 4kA^4$. Similarly, consider each of the m summands of $\Delta_{\text{quadr}}^{1:1}$. Compared to its counterpart in $\Delta_{\text{quadr}}^{\min}$, it is greater by at most

$(w_\gamma - a_{\max})a_{\max} < 4kA^4$. Combining all $2m$ summands of both of these sums, we have at most $8mkA^4$. This in turn is at most $0.08kA^5$ because without loss of generality, we can assume that $A \geq 100m$ as otherwise we could scale up the items of the Balanced Bin Packing instance by a factor of 100.

So in total, the value of σ is greater by at least $0.49k^2A^6$ minus at most $0.08kA^5$ and thus cannot attain the target objective value, so this case also does not occur. □

Corollary 4.25 *$R|HM|\sum w_jC_j$ is NP-hard already with 7 job types and* $\operatorname{rank}(\mathbf{p}) = 2$.

Corollary 4.26 *$R||\sum w_jC_j$ is* W[1]*-hard parameterized by the number of job types, even if* $\mathbf{n}$ *and* $\mathbf{p}$ *are given in unary and* $\operatorname{rank}(\mathbf{p}) = 2$.

5 Open problems

5.1 W[1]-hardness of Cutting Stock Parameterized by *k* for Unary Multiplicities

There is a pattern to the results we show: NP-hardness of scheduling problems goes hand in hand with W[1]-hardness parameterized by the number of job types for unary inputs. One may thus suspect a similar behavior for Cutting Stock once we have shown its NP-hardness for few item types (Theorem 4.17). The purpose of this short subsection is to show that if Cutting Stock is in fact W[1]-hard parameterized by the number of item types k for unary item multiplicities, then this will require substantially new ideas.

Specifically, in the proof of Theorem 4.17 we were able to construct a Cutting Stock instance which uses each bin type exactly once. We will now show that, with only unary data, this is impossible to enforce. This is because there is always an optimal solution using $\mathcal{O}(k\log(km))$ bin types. (On the other hand, our result does not apply to inputs with unary item sizes but binary item multiplicities; see the discussion below.) We will use the following structural result:

Proposition 5.27 (Aliev et al., 2018, Theorem 1) *Let $A \in \mathbb{Z}^{r\times t}$, $\mathbf{b} \in \mathbb{Z}^t$, $\mathbf{w} \in \mathbb{Z}^t$. There exists an optimal solution $\mathbf{z}^*$ to the problem $\min\{\mathbf{wx} \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}, \mathbf{x} \in \mathbb{Z}^t\}$ satisfying $|\operatorname{supp}(\mathbf{z}^*)| \leq 2r\log(2\sqrt{r}\|A\|_\infty)$ where $\operatorname{supp}(\mathbf{z}^*)$ is the* support *of the vector $\mathbf{z}^*$, which is the set of nonzero entries of the vector.*

Our result now reads as follows:

Lemma 5.28 *Let I be a* Cutting Stock *instance with $\|\mathbf{n}\|_\infty \leq \operatorname{poly}(m)$. Then, there exists an optimal solution using at most $\mathcal{O}(k\log(mk))$ bin types.*

Proof Our goal is to model I as an ILP with k rows and with $\|A\|_\infty \leq \operatorname{poly}(m)$. The result then follows straightforwardly by Proposition 5.27.

Define $\mathcal{C}_i = \{\mathbf{f} \in \mathbb{N}^k \mid \mathbf{pf} \leq s_i\}$, $i \in [m]$, to be the set of all possible configurations of a bin of type i. We call $\mathbf{f} \in \mathcal{C}_i$ a *configuration of type i*, and if a bin of type i in a solution of I contains f_j items of type j for each $j \in [k]$, we say that this bin has configuration $\mathbf{f}$. The cost of each configuration of type i is c_i. Construct A to be the matrix whose columns are first all members of $\mathcal{C}_1$, then all members of $\mathcal{C}_2$, and so on, up to $\mathcal{C}_m$. Construct an objective function $\mathbf{w}$ such that it first contains $|\mathcal{C}_1|$ copies of c_1, then $|\mathcal{C}_2|$ copies of c_2, and so on, up to $|\mathcal{C}_m|$ copies of c_m. To bound $\|A\|_\infty$, notice that each item can appear in any configuration at most $\|\mathbf{n}\|_\infty$ many times, which is bounded by $\operatorname{poly}(m)$ by our assumption. Hence, every configuration $\mathbf{f}$ is bounded by $\operatorname{poly}(m)$ in ℓ_∞-norm, and so is every column of A.

Next, any solution $\mathbf{x}$ satisfying $A\mathbf{x} = \mathbf{n}$ corresponds to a solution of Cutting Stock as follows. The value of a variable $x_\mathbf{f}$ for a configuration $\mathbf{f}$ of type i encodes that the solution contains $x_\mathbf{f}$ bins of type i with a configuration $\mathbf{f}$. By the definition of $\mathcal{C}_i$, this configuration of items indeed fits in a bin of type i, and by the constraints $A\mathbf{x} = \mathbf{n}$, each item belongs to one (and exactly one) bin. Moreover, $\mathbf{wx}$ is exactly the cost of this solution since we pay c_i for each configuration of type i. Thus, an optimum of $\min \mathbf{wx} : A\mathbf{x} = \mathbf{n}, \mathbf{x} \in \mathbb{N}^{|\mathcal{C}_1|+\cdots+|\mathcal{C}_m|}$ encodes an optimum of I.

By Proposition 5.27, there is an optimum with small support, meaning using at most $\mathcal{O}(k\log(mk))$ distinct configurations, and thus clearly at most this many bin types. □

Symmetrically, if $\mathbf{s}$ (and thus $\mathbf{p}$) are bounded by $\operatorname{poly}(m)$, it is clear that any configuration will contain at most $\operatorname{poly}(m)$ items and we get the same bound on the number of bin types necessary. Curiously, we do not see how to achieve a similar result using only a bound on $\mathbf{p}$: In the context of scheduling, we would solve a corresponding LP relaxation and then apply a proximity theorem to obtain an instance with reduced bounds and this would allow us to do some preprocessing, resulting in a bound on $\|A\|_\infty$ as above. However, because the objective function in Cutting Stock is to use as few bins as possible, this results in a different LP formulation where the proximity approach no longer works.

5.2 Other open problems

We conclude with a few interesting questions raised by our results:

- We have shown that $Q|HM|C_{\max}$ and $R|HM|C_{\max}$ are NP-hard with 6 and 4 job types, respectively. Furthermore, we have shown that CUTTING STOCK is NP-hard for 8 item types. What is the complexity for smaller

numbers of job/item types? We are not aware of any positive result except for $Q|HM|C_{\max}$ with 2 job types (McCormick et al., 2001).

- Recall the question whether $P|HM|C_{\max}$ parameterized by the number of job types k is in FPT or not. Our results provide some guidance for how one could use the interplay of high multiplicity of jobs and large job sizes to show hardness.
- We have not yet investigated jobs with release times and due dates and minimization of makespan, weighted flow time, or weighted tardiness, already on one machine. The work of Knop et al. (2019) shows that, for example, $1|r_j, d_j|\{C_{\max}, \sum w_j F_j, \sum w_j T_j\}$ parameterized by the number of job types k is in XP when $p_{\max}$ is polynomially bounded. Is it FPT or W[1]-hard?

Acknowledgements We thank the organizers of the HOMONOLO 2019 workshop for providing a warm and stimulating research environment, which gave birth to the initial ideas of this paper. Martin Koutecký was partially supported by Charles University project UNCE 24/SCI/008 and by the project 24-10306 S of GA ČR. Johannes Zink was partially supported by DFG project Wo758/11-1.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Aliev, I., Loera, J. A. D., Eisenbrand, F., Oertel, T., & Weismantel, R. (2018). The support of integer optimal solutions. *SIAM Journal on Optimization, 28*(3), 2152–2157. https://doi.org/10.1137/17M1162792

Berndt, S., Jansen, K., & Klein, K. (2021). New bounds for the vertices of the integer hull. In: *Proc. SOSA 2021*, pp. 25–36. SIAM, Philadelphia. https://doi.org/10.1137/1.9781611976496.3

Bhaskara, A., Krishnaswamy, R., Talwar, K., & Wieder, U. (2013). Minimum makespan scheduling with low rank processing times. In: Proc. SODA 2013, pp. 937–947. SIAM, Philadelphia. https://doi.org/10.1137/1.9781611973105.67

Chen, L., Marx, D., Ye, D., & Zhang, G. (2017). Parameterized and approximation results for scheduling with a low rank processing time matrix. In: Proc. STACS 2017. LIPIcs, vol. 66, pp. 22–12214. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dagstuhl. https://doi.org/10.4230/LIPIcs.STACS.2017.22

Chen, L., Jansen, K., & Zhang, G. (2018). On the optimality of exact and approximation algorithms for scheduling problems. *Journal of Computer and System Sciences, 96*, 1–32. https://doi.org/10.1016/j.jcss.2018.03.005

Conforti, M., Cornuéjols, G., & Zambelli, G. (2014). Integer Programming. Graduate Texts in Mathematics, vol. 271. Springer, Cham. https://doi.org/10.1007/978-3-319-11008-0

Cslovjecsek, J., Eisenbrand, F., Hunkenschröder, C., Rohwedder, L., & Weismantel, R. (2021). Block-structured integer and linear programming in strongly polynomial and near linear time. In *Proc. SODA 2021*, pp 1666–1681. SIAM, Philadelphia. https://doi.org/10.1137/1.9781611976465.101

Cygan, M., Fomin, F., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., & Saurabh, S. (2015). *Parameterized algorithms*. Springer. https://doi.org/10.1007/978-3-319-21275-3

Eisenbrand, F., Hunkenschröder, C., Klein, K., Koutecký, M., Levin, A., & Onn, S. (2019). An algorithmic theory of integer programming. CoRR arXiv:1904.01361 preprint

Frank, A., & Tardos, É. (1987). An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica, 7*(1), 49–65. https://doi.org/10.1007/bf02579200

Gilmore, P. C., & Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research, 9*(6), 849–859. https://doi.org/10.1287/opre.9.6.849

Goemans, M.X., & Rothvoß, T. (2014). Polynomiality for bin packing with a constant number of item types. In: *Proc. SODA 2014*, pp. 830–839. SIAM, Philadelphia. https://doi.org/10.1137/1.9781611973402.61

Goemans, M. X., & Rothvoß, T. (2020). Polynomiality for bin packing with a constant number of item types. *Journal of the ACM, 67*(6), 38–13821. https://doi.org/10.1137/1.9781611973402.61

Hermelin, D., Mnich, M., & Omlor, S. (2019). Single machine batch scheduling to minimize the weighted number of tardy jobs. CoRR arXiv:1911.12350. preprint

Hermelin, D., Karhi, S., Pinedo, M., & Shabtay, D. (2021). New algorithms for minimizing the weighted number of tardy jobs on a single machine. *Annals of Operations Research, 298*(1), 271–287. https://doi.org/10.1007/s10479-018-2852-9

Hermelin, D., Pinedo, M., Shabtay, D., & Talmon, N. (2019). On the parameterized tractability of single machine scheduling with rejection. *European Journal of Operational Research, 273*(1), 67–73. https://doi.org/10.1016/j.ejor.2018.07.038

Jansen, K. (2017). New algorithmic results for bin packing and scheduling. In: Proc. CIAC 2017, pp. 10–15. Springer, Cham. https://doi.org/10.1007/978-3-319-57586-5_2

Jansen, K., Klein, K.-M., Maack, M., & Rau, M. (2018). Empowering the configuration-IP-new PTAS results for scheduling with setups times. In: Proc. ITCS 2019, pp. 44–14419. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl. https://doi.org/10.1007/s10107-021-01694-3

Jansen, K., Lassota, A., & Maack, M. (2020). Approximation algorithms for scheduling with class constraints. In: Proc. SPAA 2020, pp. 349–357. ACM, New York. https://doi.org/10.1145/3350755.3400247

Jansen, K., & Klein, K. (2020). About the structure of the integer cone and its application to bin packing. *Mathematics of Operations Research, 45*(4), 1498–1511. https://doi.org/10.1137/1.9781611974782.103

Jansen, K., Kratsch, S., Marx, D., & Schlotter, I. (2013). Bin packing with fixed number of bins revisited. *Journal of Computer and System Sciences, 79*(1), 39–49. https://doi.org/10.1016/j.jcss.2012.04.004

Jansen, K., Lassota, A., & Rohwedder, L. (2020). Near-linear time algorithm for n-fold ILPs via color coding. *SIAM Journal on Discrete Mathematics, 34*(4), 2282–2299. https://doi.org/10.1137/19M1303873

Knop, D., & Koutecký, M. (2022). Scheduling kernels via configuration LP. In: Proc. ESA 2022, pp. 73–17315. Schloss Dagstuhl – Leibniz-

Zentrum für Informatik, Dagstuhl. https://doi.org/10.4230/LIPIcs.ESA.2022.73

Knop, D., Koutecký, M., Levin, A., Mnich, M., & Onn, S. (2019). Multitype integer monoid optimization and applications. CoRR arXiv:1909.07326. preprint

Knop, D., & Koutecký, M. (2018). Scheduling meets *n*-fold integer programming. *Journal of Scheduling, 21*(5), 493–503. https://doi.org/10.1007/s10951-017-0550-0

Knop, D., Koutecký, M., Levin, A., Mnich, M., & Onn, S. (2023). High-multiplicity n-fold IP via configuration LP. *Mathematical programming, 200*(1), 199–227. https://doi.org/10.1007/S10107-022-01882-9

Lawler, E., Lenstra, J.K., Kan, A.R., & Shmoys, D. (1993). Sequencing and scheduling: Algorithms and complexity. In: Graves, S.C., Rinnooy Kan, A.H.G., Zipkin, P.H. (eds.) Logistics of Production and Inventory. Handbooks in Operations Research and Management Science, vol. 4, pp. 445–522. North-Holland, Amsterdam. https://doi.org/10.1016/S0927-0507(05)80189-6

Levin, A. (2022). Approximation schemes for the generalized extensible bin packing problem. *Algorithmica, 84*(2), 325–343. https://doi.org/10.1007/s00453-021-00895-8

McCormick, T., Smallwood, S., & Spieksma, F. (2001). A polynomial algorithm for multiprocessor scheduling with two job lengths. *Mathematics of Operations Research, 26*(1), 31–49. https://doi.org/10.1287/moor.26.1.31.10590

Mnich, M., & Bevern, R. (2018). Parameterized complexity of machine scheduling: 15 open problems. *Computers and Operations Research, 100*, 254–261. https://doi.org/10.1016/j.cor.2018.07.020

Mnich, M., & Wiese, A. (2015). Scheduling and fixed-parameter tractability. *Mathematical Programming, 154*(1–2), 533–562. https://doi.org/10.1007/s10107-014-0830-9

Smith, W. E. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly, 3*(1–2), 59–66. https://doi.org/10.1002/nav.3800030106