Complexity of Scheduling Few Types of Jobs on Related and Unrelated Machines

Martin Koutecký
Computer Science Institute, Faculty of Mathematics and Physics,
Charles University, Prague, Czech Republic
koutecky@iuuk.mff.cuni.cz

Johannes Zink
Institut für Informatik, Universität Würzburg, Germany
johannes.zink@uni-wuerzburg.de

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 either the largest job size $p_{\max}$, or the number of jobs n are 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 speeds are polynomial in $|I|$; the same holds for $R|HM|C_{\max}$ (without 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.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Theory of computation → Scheduling algorithms

Keywords and phrases Scheduling, cutting stock, hardness, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2020.18

Related Version A full version of the paper is available at https://arxiv.org/abs/2009.11840.

Funding *Martin Koutecký*: M. Koutecký was partially supported by Charles University project UNCE/SCI/004, by the project 19-27871X of GA ČR, and by an Israel Science Foundation grant 308/18.

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. We also thank the anonymous reviewers for their helpful remarks.

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 $p_{\max} = \max_j p_j$. Goemans and Rothvoss [10] 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.

Since 2014, considerable attention has been given to studying the complexity of various scheduling problems in the regime with few job types [15, 22, 20, 21, 3, 11, 13, 26, 14, 12], and similar techniques have been used to obtain approximation algorithms [16, 24, 18]. 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 shortly as $P||C_{\max}$, $Q||C_{\max}$, and $R||C_{\max}$ [23], respectively, with the high-multiplicity variant being $P|HM|C_{\max}$ and analogously for the other models. 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 $\mathbf{s} = (s_1, \ldots, s_m)$. Hence, the rank of the job sizes matrix has been studied [1, 3, 2] as a helpful measure of complexity of an $R||C_{\max}$ instance: intuitively, the smaller the rank, the closer is the instance to $Q||C_{\max}$. We answer the question above:

▶ **Theorem 1.** *$Q|HM|C_{\max}$ 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 a different capacity and cost, 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 [9]. 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 2.** *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

[1] The complexity stated in [10] is $(\log \max C_{\max}, n)^{f(k)} \operatorname{poly}\log n$, but a close inspection of their proof reveals that **a)** the dependence on n is unnecessary, and **b)** it is possible to use a better bound on the number of vertices of a polytope and obtain the complexity stated here.

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 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 [6].

The hard instance I from Theorem 1 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 $\lceil \log_2 p \rceil + \lceil \log_2 q \rceil$, a $Q||C_{\max}$ instance with $\mathbf{s}$ of polynomial length might translate to an $R||C_{\max}$ instance with $\mathbf{p}$ of exponential length. 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.** *$\{R,Q\}|HM|C_{\max}$ 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 [7] is required:

▶ **Theorem 4.** *$\{R,Q\}|HM|C_{\max}$ can be solved in time $p_{\max}^{\mathcal{O}(k^2)} m \log m \log^2 n$, 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 $\mathbf{n}$, $\mathbf{p}$, and $\mathbf{s}$ are encoded in unary, and for $R|HM|C_{\max}$ even when the rank of $\mathbf{p}$ is 2:

▶ **Theorem 5.** *$X||C_{\max}$ is W[1]-hard parameterized by the number of job types with*
(a) *$X = Q$ and $\mathbf{n}$, $\mathbf{p}$, and $\mathbf{s}$ given in unary.*
(b) *$X = R$ and $\mathbf{n}$ and $\mathbf{p}$ given in unary and $\mathrm{rank}(\mathbf{p}) = 2$.*

We use a result of Jansen et al. [17] 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 given in unary. 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 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 an $R|HM|C_{\max}$ instance of makespan T in which optimal solutions are in bijection with optimal packings of the encoded Balanced Bin Packing instance. Our $R|HM|C_{\max}$ instance uses one 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, and all other job types have sizes independent of which machine they run on. Since the capacity of a machine exactly corresponds to its speed, omitting those "blocker" jobs and setting the machine speeds gives a hard instance for $Q|HM|C_{\max}$.

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 $(\log p_{\max})^{f(k)} \operatorname{poly}\log n \leq g(k) \cdot p_{\max}^{o(1)} \operatorname{poly}\log n$ [6, Exercise 3.18]. We answer the remaining question whether the problem is in FPT also when all jobs are given explicitly:

▶ **Theorem 6.** *$P||C_{\max}$ is FPT parameterized by k.*

This result partially answers [25, Question 5], which asks for an FPT algorithm for $P|HM|C_{\max}$. Obtaining this answer turns out to be surprisingly easy: we reduce the job sizes by a famous algorithm of Frank and Tardos [8] and then apply the algorithm of Goemans and Rothvoss [10], 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 either the largest job or the number of jobs are not too large. 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.

Table 1 Overview of the computational hardness of $\{P, Q, R\}|\{\_, HM\}|\{C_{\max}, \ell_2, \sum w_jC_j\}$ relative to the number of job types k.

	$P\|\|\dots$	$Q\|\|\dots$		$R\|\|\dots$		$P\|HM\|\dots$	$Q\|HM\|\dots$	$R\|HM\|\dots$
$C_{\max}$	FPT (Thm. 6)	XP (Theorem 3)	W[1]-hard (Thm. 5)	XP (Theorem 3)	W[1]-hard (Thm. 5)	poly. time for const. k ([10])	NP-hard for $k \geq 6$ (Thm. 1)	NP-hard for $k \geq 4$ (Thm. 17)
ℓ_2	?		W[1]-hard (Cor. 23)		W[1]-hard (Cor. 23)	?	NP-hard for $k \geq 6$ (Cor. 22)	NP-hard for $k \geq 7$ (Cor. 22)
$\sum w_jC_j$	?		?		W[1]-hard (Cor. 27)	?	?	NP-hard for $k \geq 7$ (Cor. 26)

Other Objectives. Besides minimum makespan, two important scheduling objectives are minimization of the sum of weighted completion times, denoted $\sum w_jC_j$, 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 $\|(L_1, \dots, L_m)\|_2 = \sqrt{\sum_{i=1}^m L_i^2}$. 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, denoted C_j, is the time it finishes its execution in a schedule. In the $R|HM|\sum w_jC_j$ problem, each job is additionally given a weight w_j, and the task is to minimize $\sum w_jC_j$.

We show that the hard instance for $R|HM|C_{\max}$ is also hard for ℓ_2, and with the right choice of weights is also hard for $\sum w_jC_j$. We also obtain hardness of $Q|HM|\ell_2$ by a different and more involved choice of speeds, but the case of $Q|HM|\sum w_jC_j$ 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 $R|HM|\sum w_jC_j$ because the value $\sum w_jC_j$ is proportional to the load vector plus other terms, and we need to bound those remaining terms (Lemma 24) in order to transfer the argument from ℓ_2 to $\sum w_jC_j$. 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 hard instance of $Q|HM|C_{\max}$.

Proofs of statements marked with "$\star$" are available in the full version (`https://arxiv.org/abs/2009.11840`). 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 $\mathbf{xy}$ [4]. We use $\log := \log_2$, i.e., all our logarithms are base 2. For $n, m \in \mathbb{N}$, we write $[n, m] = \{n, n+1, \dots, m\}$ and $[n] = [1, n]$.

Makespan Minimization on Unrelated Machines ($R|HM|C_{\max}$)

Input: n jobs of k types, job multiplicities $n_1, \dots, n_k$, 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 $\mathbf{p}^i = (p^i_1, \dots, p^i_k) \in (\mathbb{N} \cup \{+\infty\})^{k \cdot m}$ 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.

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: we denote by $R||C_{\max}$ the problem where jobs are given explicitly, and by $R|HM|C_{\max}$ the problem defined above; see also the discussion in [21].

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

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, 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 $\mathbf{s} = (s_1, \dots, s_m)$, 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 $P||C_{\max}$. Its decision variant $P||C_{\max}$ is equivalent to Bin Packing:

Bin Packing

Input: n items of sizes $a_1, \dots, a_n$, 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 all $a_1, \dots, a_n$ are encoded in unary, or, equivalently, $a_{\max} = \max_i a_i$ 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 $\sum_i a_i$ is equal to $k \cdot B$, which means that if an instance has a packing, then each bin is used fully.

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 [20] 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, $E_1^i \in \mathbb{Z}^{r\times t}$ and $E_2^i \in \mathbb{Z}^{s\times t}$ for all $i \in [N]$, are integer matrices, and f is some separable convex function. Specifically for $R||C_{\max}$, $f \equiv 0$, the matrices corresponding to equations (1)–(2) are $E_1^i = I$ and $E_2^i = \mathbf{p}^i$, for each $i \in [m]$, $\mathbf{b} = (\mathbf{n}, T, \dots, T)$ is an $r + Ns = (k + m)$-dimensional vector, and $\mathbf{l} = \mathbf{0}$ and $\mathbf{u} = (\mathbf{n}, \mathbf{n}, \dots, \mathbf{n})$ are $Nt = (mk)$-dimensional vectors. We note that N-fold IP formulations are also known for $R|HM|\{\ell_2, \sum w_j C_j\}$ [21, 20].

3.1 Large Lengths, Polynomial Multiplicities

A simple dynamic programming algorithm gives:

▶ **Theorem 3** ($\star$). *$\{R, Q\}|HM|\{C_{\max}, \ell_2, \sum w_j C_j\}$ 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.*

Theorem 3 (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 need the following result:

▶ **Proposition 7** (Frank and Tardos [8]). *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 8.** *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 7 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 7. ◀

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

▶ **Proposition 9** ([19, 5, 7])**.** *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 for $C_{\max}$. By Lemma 8, 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 9 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 11. ◀

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

▶ **Theorem 6.** *$P||C_{\max}$ is FPT parameterized by k.*

Proof. Apply Lemma 8 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 [10] have shown that $P||C_{\max}$ with k job types can be solved in time $(\log p_{\max})^{2^{\mathcal{O}(k)}} \operatorname{poly} \log n$. 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., $f(k) \operatorname{poly}(n)$ for some computable f), we use a simple observation [6, 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 10. The algorithm of [10] 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 6 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.

A straightforward adaptation of the proof of Lemma 8 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 11.** *Given an N-fold IP instance 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 $\|\bar{E}_2^i, \bar{\mathbf{b}}^i\|_\infty \leq 2^{\mathcal{O}(t^3)} M^{\mathcal{O}(t^2)}$. ◀

3.2 Polynomial Lengths, 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 9:

▶ **Theorem 4.** *$\{R,Q\}|HM|\{C_{\max},\ell_2,\sum w_jC_j\}$ can be solved in time $p_{\max}^{\mathcal{O}(k^2)} m\log m\log^2 n$.*

To obtain a result like this one can first solve the LP relaxation of (1)–(2), and then use a "proximity theorem" to show that some integral optimum is at distance at most $p_{\max}^{\mathcal{O}(k)}\cdot m$ [7, Theorem 59] from any optimum of the LP relaxation. This yields an $\{R,Q\}|HM|\{C_{\max},\ell_2,\sum w_jC_j\}$ instance where roughly $p_{\max}^k\cdot m$ jobs are left to be scheduled and which can be solved using Theorem 3. 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 $\sum w_jC_j$, the modification is analogous but slightly more complicated; the approach is identical to the one described by Knop and Koutecký [20].

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 $p_{\max}\le k$, the best algorithm for $Q|HM|C_{\max}$ has a dependence of k^{k^2} [22, 20].

4 Hardness

4.1 Reducing Bin Packing to Balanced Bin Packing

▶ **Lemma 12.** *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 $n',k',B',a'_{\max}$ 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, assume for the sake of contradiction that the Balanced Bin Packing instance has a solution, but the original Bin Packing instance does not. Consider a solution of Balanced Bin Packing, 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– a contradiction.

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 iff the Bin Packing instance was. ◀

▶ **Corollary 13.** *Balanced Bin Packing is NP-hard, even for tight instances.*

▶ **Corollary 14.** *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 hard 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 specific schedule, which we call henceforth *perfect*, is optimal. In a perfect schedule, M_i gets 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 for some $j \in [k]$. 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. We set, for $j \in [k]$,

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

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

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

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

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

[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.

▶ **Lemma 15.** *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 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 hard instance I above. It remains to prove the correctness of our reduction. 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, to machine M_i 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. According to equations (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 and we have to place m such 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 job of type β_j to a machine, we can execute on this machine at most A jobs of type $\alpha_{j'}^{\times}$ for any j'. In particular, observe that even on a machine that executes β_1, which is the smallest of the β-type jobs, we cannot add $A + 1$ jobs of type α_k^0, which is the smallest of the $\alpha_j^{\times}$ job types, without exceeding $T + \max_i a_i$.

For each $j \in [k]$, there are Am/k jobs of type $\alpha_j^{\times}$. Thus, there are exactly A α-type jobs on each machine from $\mathcal{M}_j$. Observe that on a machine from $\mathcal{M}_j$, we cannot use a job $\alpha_{j'}$, where $j' < j$, as this would exceed $T + a_i$. Therefore, we have to execute 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 have 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[3] $a_{\max} < A < p_t$ for any job type t, so no other job can be scheduled. Consider the sizes of the jobs that have to be executed on a machine. There can be at most a_i jobs of type α_j^1 on each machine M_i. Hence we have, for each $j \in [k]$,

$$A/k \le \sum_{M_i \in \mathcal{M}_j} a_i \tag{5}$$

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

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

[3] 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.

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 Equation (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. ◀

We can easily adjust our 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.

▶ **Lemma 16** (⋆). *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) *in any feasible schedule, all machines finish precisely by time $T_R = 7kA^3$, and*

(e) *the job sizes matrix $\mathbf{p}$ has rank 2.*

Applying the reductions of Lemmas 15 and 16 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 1.** *$Q|HM|C_{\max}$ is NP-hard already with 6 job types.*

▶ **Theorem 17** (⋆). *$R|HM|C_{\max}$ is NP-hard already with 4 job types and with $\mathbf{p}$ of rank 2.*

The complexity of $Q|HM|C_{\max}$ ($R|HM|C_{\max}$) with less than 6 (4) job types remains open.

From Lemmas 15 and 16 and the hardness of Corollary 14, we also get our main result:

▶ **Theorem 5.** *$X||C_{\max}$ is W[1]-hard parameterized by the number of job types with*

(a) *$X = Q$ and $\mathbf{n}$, $\mathbf{p}$, and $\mathbf{s}$ given in unary.*

(b) *$X = R$ and $\mathbf{n}$ and $\mathbf{p}$ given in unary and $\operatorname{rank}(\mathbf{p}) = 2$.*

4.3 NP-hardness of 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.

The difficulty in transferring hardness from $Q|HM|C_{\max}$ to Cutting Stock is in enforcing that each bin type is used exactly once.

▶ **Lemma 18.** *$Q|HM|C_{\max}$ 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 $2^0, 2^1, \dots, 2^{m-1}$. 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 18 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 18 yields:

▶ **Theorem 2.** *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 the speed s_i of machine M_i depended 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 a different speed 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

$$s_i = \left\lceil (T + a_{\max})\sqrt{T + a_i} \right\rceil . \tag{6}$$

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 $\mathbf{x}^i = (x_1^i, \dots, x_\tau^i)$ is the vector of job multiplicities scheduled to machine M_i, and τ is the number of job types.

▶ **Lemma 19.** *The hardness instance I with modified s_i is also hard for $Q|HM|\ell_2^2$ with 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 for each $i \in [m]$. 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 15 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 $(T+a_1, \ldots, T+a_m)$ of a perfect schedule (as it appears in Case 1) 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 with the smallest deviation (minimizing $\Delta_i = |\bar{L}_i - (T+a_i)|$) and move Δ_i integral units of load from it or to it (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, 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

$$\left(\frac{T+a_i+z_i}{s_i}\right)^2 + \left(\frac{T+a_j-z_j}{s_j}\right)^2 < \left(\frac{T+a_i+z_i+r}{s_i}\right)^2 + \left(\frac{T+a_j-z_j-r}{s_j}\right)^2$$

$$\Leftrightarrow \qquad s_i^2\left(2r(T+a_j-z_j)-r^2\right) < s_j^2\left(2r(T+a_i+z_i)+r^2\right)$$

$$\Leftrightarrow \qquad \frac{s_i^2}{s_j^2} < \frac{2(T+a_i+z_i)+r}{2(T+a_j-z_j)-r}\ . \tag{7}$$

Next, we analyze the machine speed s_i as defined in equation (6). Recall that we scale up $\sqrt{T+a_i}$ by a sufficiently large factor b to make each $\left\lceil\sqrt{T+a_i}\right\rceil$ different from $\left\lceil\sqrt{T+a_i-1}\right\rceil$ If the difference between $\sqrt{T+a_i}$ and $\sqrt{T+a_i-1}$ is at least d, then it must hold that

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

We have chosen $b = T + a_{\max}$, since $x > 1/(\sqrt{x} - \sqrt{x-1})$ for $x \geq 4$. Hence, we conclude

$$\left\lceil (T + a_{\max})\sqrt{T + a_i} \right\rceil < (T + a_{\max})\sqrt{T + a_i + 1} \ . \tag{8}$$

With this inequality in hand, we finally show the correctness of inequality (7):

$$\frac{s_i^2}{s_j^2} = \frac{\left\lceil (T + a_{\max})\sqrt{T + a_i} \right\rceil^2}{\left\lceil (T + a_{\max})\sqrt{T + a_j} \right\rceil^2} \overset{(8)}{<} \frac{(T + a_{\max})^2(T + a_i + 1)}{(T + a_{\max})^2(T + a_j)} = \frac{2(T + a_i) + 2}{2(T + a_j)}$$
$$\overset{(r \geq 1)}{\leq} \frac{2(T + a_i) + 2r}{2(T + a_j)} \overset{(a_i \geq a_j)}{<} \frac{2(T + a_i) + r}{2(T + a_j) - r} \leq \frac{2(T + a_i + z_i) + r}{2(T + a_j - z_j) - r}$$ ◀

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

▶ **Lemma 20.** *The hardness instance I_R is hard for $R|HM|\ell_2^2$ 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 for each $i \in [m]$. 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 $mT_R^2 = 49mk^2A^6$. 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 16, 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 16, 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. [20, 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 21, which we prove in the full version. Again, it contradicts our assumption of σ having objective value at most mT_R^2. So this case can also not occur.

▷ Claim 21 (⋆). 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

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

The following corollaries follow immediately from Lemmas 19 and 20; as before, it is likely that one might improve this to 4 job types.

▶ **Corollary 22.** *$X|HM|\ell_2$ 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 23.** *$X||\ell_2$ is W[1]-hard parameterized by the number of job types with*
(a) *$X = Q$ and $\mathbf{n}$, $\mathbf{p}$, and $\mathbf{s}$ given in unary.*
(b) *$X = R$ and $\mathbf{n}$ and $\mathbf{p}$ 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 16. 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 [27]. 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 20) because it is known that $\sum w_jC_j$ and ℓ_2 are often (not always) closely related. However, because the size of a job of type γ depends both on j and the machine M_i, yet its weight only depends on j, it is impossible to express an exact objective value of the perfect schedule from the previous sections. This would make the argument of an analogue of Case 2a of Lemma 20 invalid and a no-instance of Balanced Bin Packing might reduce to a yes-instance of $R||\sum w_jC_j$. 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. However, the contribution of jobs of type γ depends on the machine, while its weight is 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 20. For Case 2b, we will exploit the claim in the proof of Lemma 20 once again and combine it with a gap argument (Lemma 24).

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

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

▶ **Lemma 24** (⋆). *Let σ be any schedule of the weighted hardness instance, let $(L_1, L_2, \ldots, L_m)$ be its load vector, and $\mathcal{L} := \frac{1}{2}\left(\sum_{i=1}^m {L_i}^2\right)$. Let $\Gamma = \frac{1}{2k}\sum_{j=1}^k \left(Aw_{\alpha_j^0}^2 + (m-1)Aw_{\alpha_j^1}^2 + mw_{\beta_j}^2\right)$, $\Delta_{\text{linear}}^{1:1} = \frac{1}{2}\sum_{i=1}^m p_\gamma^i w_\gamma$, $\Delta_{\text{quadr}}^{1:1} = \frac{1}{2}\sum_{i=1}^m p_\gamma^i \cdot a_i$, $\Delta^{1:1} = \Delta_{\text{linear}}^{1:1} + \Delta_{\text{quadr}}^{1:1}$, $\Delta_{\text{linear}}^{\min} = m(w_\gamma - a_{\max})w_\gamma$, and $\Delta_{\text{quadr}}^{\min} = m(w_\gamma - a_{\max})a_{\max}$.*

(a) *The value of σ under $\sum w_jC_j$ is at least $\mathcal{L} + \Gamma + \Delta_{\text{linear}}^{\min} + \Delta_{\text{quadr}}^{\min}$.*

(b) *If σ schedules one γ job per machine, then the value σ under $\sum w_jC_j$ is $\mathcal{L} + \Gamma + \Delta^{1:1}$.*

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

▶ **Lemma 25** (⋆). *The weighted hardness instance I_R^w is hard for $R|HM|\sum w_jC_j$.*

▶ **Corollary 26.** *$R|HM|\sum w_jC_j$ is NP-hard already with 7 job types and* $\text{rank}(\mathbf{p}) = 2$.

▶ **Corollary 27.** *$R||\sum w_jC_j$ is* W[1]*-hard parameterized by the number of job types, even if* $\mathbf{n}$ *and* $\mathbf{p}$ *are given in unary and* $\text{rank}(\mathbf{p}) = 2$.

5 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. What is the complexity for smaller numbers of job types? We are not aware of any positive result about either problem, including Cutting Stock, even for 2 job/item types.
- 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.
- Is Cutting Stock W[1]-hard when the input data is given in unary?
- We haven't 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. [22] 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?

References

1 Aditya Bhaskara, Ravishankar Krishnaswamy, Kunal Talwar, and Udi Wieder. Minimum makespan scheduling with low rank processing times. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 937–947. SIAM, 2013.

2 Lin Chen, Klaus Jansen, and Guochuan Zhang. On the optimality of exact and approximation algorithms for scheduling problems. *Journal of Computer and System Sciences*, 96:1–32, 2018.

3 Lin Chen, Dániel Marx, Deshi Ye, and Guochuan Zhang. Parameterized and approximation results for scheduling with a low rank processing time matrix. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, pages 22:1–22:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.STACS.2017.22.

4 Michele Conforti, Gérard Cornuéjols, Giacomo Zambelli, et al. *Integer programming*, volume 271. Springer, 2014.

5 Jana Cslovjecsek, Friedrich Eisenbrand, and Robert Weismantel. N-fold integer programming via LP rounding. *arXiv preprint*, 2020. arXiv:2002.07745.

6 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

7 Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Asaf Levin, and Shmuel Onn. An algorithmic theory of integer programming. *CoRR*, 2019. arXiv:1904.01361.

8 András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.

9 P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Oper. Res.*, 9:849–859, 1961.

10 Michel X. Goemans and Thomas Rothvoß. Polynomiality for bin packing with a constant number of item types. In *Proc. SODA 2014*, pages 830–839, 2014.

11 Danny Hermelin, Shlomo Karhi, Michael Pinedo, and Dvir Shabtay. New algorithms for minimizing the weighted number of tardy jobs on a single machine. *Annals of Operations Research*, pages 1–17, 2018.

12 Danny Hermelin, Matthias Mnich, and Simon Omlor. Single machine batch scheduling to minimize the weighted number of tardy jobs. *arXiv preprint*, 2019. arXiv:1911.12350.

13 Danny Hermelin, Michael Pinedo, Dvir Shabtay, and Nimrod Talmon. On the parameterized tractability of single machine scheduling with rejection. *European Journal of Operational Research*, 273(1):67–73, 2019.

14 Klaus Jansen. New algorithmic results for bin packing and scheduling. In Dimitris Fotakis, Aris Pagourtzis, and Vangelis Th. Paschos, editors, *Algorithms and Complexity*, pages 10–15, Cham, 2017. Springer International Publishing.

15 Klaus Jansen and Kim-Manuel Klein. About the structure of the integer cone and its application to bin packing. In *Proc. SODA 2017*, pages 1571–1581, 2017.

16 Klaus Jansen, Kim-Manuel Klein, Marten Maack, and Malin Rau. Empowering the configuration-IP-new PTAS results for scheduling with setups times. In *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

17 Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *Journal of Computer and System Sciences*, 79(1):39–49, 2013.

18 Klaus Jansen, Alexandra Lassota, and Marten Maack. Approximation algorithms for scheduling with class constraints. *arXiv preprint*, 2019. `arXiv:1909.11970`.

19 Klaus Jansen, Alexandra Lassota, and Lars Rohwedder. Near-linear time algorithm for n-fold ILPs via color coding. *arXiv preprint*, 2018. `arXiv:1811.00950`.

20 Dušan Knop and Martin Koutecký. Scheduling meets n-fold integer programming. *Journal of Scheduling*, 21:493–503, 2018.

21 Dusan Knop and Martin Koutecký. Scheduling kernels via configuration LP. *CoRR*, abs/2003.02187, 2020. `arXiv:2003.02187`.

22 Dušan Knop, Martin Koutecký, Asaf Levin, Matthias Mnich, and Shmuel Onn. Multitype integer monoid optimization and applications. *arXiv preprint*, 2019. `arXiv:1909.07326`.

23 Eugene L. Lawler, Jan Karel Lenstra, Alexander H. G. Rinnooy Kan, and David B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In S. C. Graves, A. H. G. Rinnooy Kan, and P. H. Zipkin, editors, *Handbooks in Operations Research and Management Science: Logistics of Production and Inventory*, volume 4, pages 445–522, Amsterdam-London-New York-Tokyo, 1993. North-Holland Publishing Company.

24 Asaf Levin. Approximation schemes for the generalized extensible bin packing problem. *arXiv preprint*, 2019. `arXiv:1905.09750`.

25 Matthias Mnich and René van Bevern. Parameterized complexity of machine scheduling: 15 open problems. *Computers & OR*, 100:254–261, 2018.

26 Matthias Mnich and Andreas Wiese. Scheduling and fixed-parameter tractability. *Mathematical Programming*, 154(1-2):533–562, 2015.

27 Wayne E. Smith. Various optimizers for single-stage production. *Naval Res. Logist. Quart.*, 3:59–66, 1956.