Closing the Gap for Makespan Scheduling via Sparsification Techniques* †

Klaus Jansen[1], Kim-Manuel Klein[2], and José Verschae[3]

1 **Department of Computer Science, University of Kiel, Kiel, Germany**
kj@informatik.uni-kiel.de
2 **Department of Computer Science, University of Kiel, Kiel, Germany**
kmk@informatik.uni-kiel.de
3 **Facultad de Matemáticas and Escuela de Ingeniería, Pontificia Universidad Católica de Chile, Santiago, Chile**
jverschae@uc.cl

Abstract

Makespan scheduling on identical machines is one of the most basic and fundamental packing problem studied in the discrete optimization literature. It asks for an assignment of n jobs to a set of m identical machines that minimizes the makespan. The problem is strongly NP-hard, and thus we do not expect a $(1+\varepsilon)$-approximation algorithm with a running time that depends polynomially on $1/\varepsilon$. Furthermore, Chen et al. [3] recently showed that a running time of $2^{(1/\varepsilon)^{1-\delta}} + \text{poly}(n)$ for any $\delta > 0$ would imply that the Exponential Time Hypothesis (ETH) fails. A long sequence of algorithms have been developed that try to obtain low dependencies on $1/\varepsilon$, the better of which achieves a running time of $2^{\tilde{O}(1/\varepsilon^2)} + O(n \log n)$ [10]. In this paper we obtain an algorithm with a running time of $2^{\tilde{O}(1/\varepsilon)} + O(n \log n)$, which is tight under ETH up to logarithmic factors on the exponent.

Our main technical contribution is a new structural result on the *configuration-IP*. More precisely, we show the existence of a highly symmetric and sparse optimal solution, in which all but a constant number of machines are assigned a configuration with small support. This structure can then be exploited by integer programming techniques and enumeration. We believe that our structural result is of independent interest and should find applications to other settings. In particular, we show how the structure can be applied to the minimum makespan problem on related machines and to a larger class of objective functions on parallel machines. For all these cases we obtain an efficient PTAS with running time $2^{\tilde{O}(1/\varepsilon)} + \text{poly}(n)$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases scheduling, approximation, PTAS, makespan, ETH

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.72

1 Introduction

Minimum makespan scheduling is one of the foundational problems in the literature on approximation algorithms [6, 7]. In the *identical machine* setting the problem asks for an assignment of a set of n jobs $\mathcal{J}$ to a set of m identical machines $\mathcal{M}$. Each job $j \in \mathcal{J}$ is

* Full version: http://arxiv.org/abs/1604.07153.

† This work was partially supported by DFG Project "Entwicklung und Analyse von effizienten polynomiellen Approximationsschemata für Scheduling- und verwandte Optimierungsprobleme," Ja 612/14-2, by FONDECYT project 3130407, and by Núcleo Milenio Información y Coordinación en Redes ICM/FIC RC130003.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 72; pp. 72:1–72:13

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

characterized by a non-negative processing time $p_j \in \mathbb{Z}_{>0}$. The load of a machine is the total processing time of jobs assigned to it, and our objective is to minimize the *makespan*, that is, the maximum machine load. This problem is usually denoted $P||C_{\max}$. It is well known to admit a *polynomial time approximation scheme* (PTAS) [9], and there has been many subsequent works improving the running time or deriving PTAS's for more general settings. The fastest PTAS for $P||C_{\max}$ achieves a running time of $2^{O(1/\varepsilon^2)\log^3(1/\varepsilon))} + O(n \log n)$ for $(1+\varepsilon)$-approximate solutions [10]. Very recently, Chen et al. [3] showed that, assuming the *exponential time hypothesis* (ETH), there is no PTAS that yields $(1+\varepsilon)$-approximate solutions for $\varepsilon > 0$ with running time $2^{(1/\varepsilon)^{1-\delta}} + \text{poly}(n)$ for any $\delta > 0$ [3].

Given a guess $T \in \mathbb{N}$ on the optimal makespan, which can be found with binary search, the problem reduces to deciding the existence of a packing of the jobs to m machines (or bins) of capacity T. If we aim for a $(1+\varepsilon)$-approximate solution, for some $\varepsilon > 0$, we can assume that all processing times are integral and T is a constant number, namely $T \in O(1/\varepsilon^2)$. This can be achieved with well known rounding and scaling techniques [1, 2, 8] which will be specified later. Let $\pi_1 < \pi_2 < \ldots < \pi_d$ be the job sizes appearing in the instance after rounding, and let b_k denote the number of jobs of size π_k. The mentioned rounding procedure implies that the number of different job sizes is $d = O((1/\varepsilon)\log(1/\varepsilon))$. Hence, for large n we obtain a highly symmetric problem where several jobs will have the same processing time. Consider the *knapsack polytope* $\mathcal{P} = \{c \in \mathbb{R}^d_{\geq 0} : \pi \cdot c \leq T\}$. A packing on one machine can be expressed as a vector $c \in Q = \mathbb{Z}^d \cap \mathcal{P}$, where c_k denotes the number of jobs of size π_k assigned to the machine. Elements in $Q = \mathbb{Z}^d \cap \mathcal{P}$ are called *configurations*. Considering a variable $x_c \in \mathbb{Z}_{\geq 0}$ that decides the multiplicity of configuration c in the solution, our problem reduces to solving the following linear integer program (ILP):

$$\text{[conf-IP]} \quad \sum_{c \in Q} c \cdot x_c = b, \tag{1}$$

$$\sum_{c \in Q} x_c = m, \tag{2}$$

$$x_c \in \mathbb{Z}_{\geq 0} \qquad \text{for all } c \in Q. \tag{3}$$

In this article we derive new insights on this ILP that help us to design faster algorithms for $P||C_{\max}$ and other more general problems. These including makespan scheduling on *related machines* $Q||C_{\max}$, and a more general class of objective functions on parallel machines. We show that all these problems admit a PTAS with running time $2^{O((1/\varepsilon)\log^4(1/\varepsilon))} + \text{poly}(n)$. Hence, our algorithm is best possible up to polylogarithmic factors in the exponent assuming ETH [3].

1.1 Literature Review

There is an old chain of approximation algorithms for $P||C_{\max}$, starting from the seminal work by Graham [6, 7]. The first PTAS was given by Hochbaum and Shmoys [9] and had a running time of $(n/\varepsilon)^{O((1/\varepsilon)^2)} = n^{O((1/\varepsilon)^2 \log(1/\varepsilon))}$. This was improved to $n^{O((1/\varepsilon)\log^2(1/\varepsilon))}$ by Leung [14]. Subsequent articles improve further the running time. In particular Hochbaum and Shmoys (see [8]) and Alon et al. [1, 2] obtain an *efficient PTAS*[1] (EPTAS) with running time $2^{(1/\varepsilon)^{\text{poly}(1/\varepsilon)}} + O(n \log n)$. Alon et al. [1, 2] consider general techniques that work for

[1] That is, a PTAS whose running time is $f(1/\varepsilon)\text{poly}(|I|)$ where $|I|$ is the encoding size of the input and f is some function.

several objective functions, including all L_p-norm of the loads and maximizing the minimum machine load.

The previously fastest PTAS for $P||C_{\max}$ achieves a running time of $2^{O((1/\varepsilon)^2 \log^3(1/\varepsilon))} + O(n \log n)$ [10]. More generally, this work gives an EPTAS for the case of related (uniform) machines, where each machine $i \in \mathcal{M}$ has a speed s_i and assigning to i job j implies a processing time of p_j/s_i. For this more general case the running time is $2^{O((1/\varepsilon)^2 \log^3(1/\varepsilon))} + \text{poly}(n)$. For the simpler case of $P||C_{\max}$, the ILP can be solved directly since the number of variables is a constant. This can be done with Lentras' algorithm [13], or even with Kannan's algorithm [12] that gives an improved running time. This technique yields a running time that is doubly exponential in $1/\varepsilon$. This was, in essence, the approach by Alon et al. [1, 2] and Hochbaum and Shmoys [8]. To lower the dependency on $1/\varepsilon$, Jansen [10] uses a result by Eisenbrand and Shmonin [4] that implies the existence of a solution x with support of size at most $O(d \log(dT)) = O((1/\varepsilon) \log^2(1/\varepsilon))$. First guessing the support and then solving the ILP with $O((1/\varepsilon) \log^2(1/\varepsilon))$ integer variables and using Kannan's algorithm yields the desired running time of $2^{O((1/\varepsilon)^2 \log^3(1/\varepsilon))} + O(n \log n)$.

The configuration ILP has recently been studied in the context of the (1-dimensional) cutting stock problem. In this case, the dimension d is constant, $T = 1$, and π is a rational vector. Moreover, π and b are part of the input. Goemans and Rothvoß [5] obtain an optimal solution in time $\log(\Delta)^{2^{O(d)}}$, where Δ is the largest number appearing in the denominator of π_k or the multiplicities b_k. This is achieved by first showing that there exists a pre-computable set $\tilde{Q} \subseteq Q$ with polynomial many elements, such that there exists a solution x that gives all but constant (depending only on d) amount of weight to $\tilde{Q}$. We remark that applying this result to a rounded instance of $P||C_{\max}$ yields a running time that is doubly exponential on $1/\varepsilon$.

1.2 Our Contributions

Our main contribution is a new insight on the structure of the solutions of [conf-IP]. These properties are specially tailored to problems in which T is bounded by a constant, which in the case of $P||C_{\max}$ can be guaranteed by rounding and scaling. The same holds for $Q||C_{\max}$ with a more complex rounding and case analysis.

We first classify configurations by their support. We say that a configuration is *simple* if its support is of size at most $\log(T+1)$, otherwise it is *complex*. Our main structural result[2] states that there exists a solution x in which all but $O(d \log(dT))$ weight is given to simple configurations, the support is bounded by $O(d \log(dT))$ (as implied by Eisenbrand and Shmonin [4]) and no complex configuration has weight larger than 1.

▶ **Theorem 1** (Thin solutions). *Assume that [conf-IP] is feasible. Then there exists a feasible solution x to [conf-IP] such that:*

1. *if $x_c > 1$ then the configuration c is simple,*
2. *the support of x satisfies $|\text{supp}(x)| \leq 4(d+1)\log(4(d+1)T)$, and*
3. *$\sum_{c \in Q_c} x_c \leq 2(d+1)\log(4(d+1)T)$, where Q_c denotes the set of complex configurations.*

[2] We remark the resemblance of this structure to the result by Goemans and Rothvoß [5]. Indeed, similarly to their result, we can precompute a subset of configurations such that all but a constant amount of weight of the solution is given to such set. In their case the set is of cardinality polynomial on the input and is constructed by covering the integral solutions of the knapsack polytope by parallelepipeds. In our case, all but $O(d \log dT)$ weight is given to simple configurations.

We call a solution satisfying the properties of the theorem *thin*. The theorem can be shown by iteratively applying a sparsification lemma that shows that if a solution gives a weight of two or more to a complex configuration, then we can replace this partial solution by two configurations with smaller support. The sparsification lemma is shown by a simple application of the pigeonhole principle. The theorem can be shown by mixing this technique with the theorem of Eisenbrand and Shmonin [4] and a potential function argument.

As an application to our main structural theorem, we derive a PTAS for $P||C_{\max}$ by first guessing the jobs assigned to complex configurations. An optimal solution for this subinstance can be derived by a dynamic program. For the remaining instance we know the existence of a solution using only simple configurations. Then we can guess the support of such solution and solve the corresponding [conf-IP] restricted to the guessed variables. The main use of having simple configurations is that we can guess the support of the solution much faster, as the number of simple configuration is (asymptotically) smaller than the total number of configurations. The complete procedure takes time $2^{O((1/\varepsilon)\log^4(1/\varepsilon))} + O(n \log n)$. Moreover, using the rounding and case analysis of Jansen [10], we derive an mixed integer linear program that can be suitably decomposed in order to apply our structural result iteratively. This yields a PTAS with a running time of $2^{O((1/\varepsilon)\log^4(1/\varepsilon))} + \text{poly}(n)$ for $Q||C_{\max}$.

Similarly, we can extend our results to derive PTAS's for a larger family of objective functions as considered by Alon et al. [1, 2]. Let ℓ_i denote the load of machine i, that is, the total processing time of jobs assigned to machine i for a given solution. Our techniques then gives a PTAS with the same running time for the problem of minimizing the L_p-norms of the loads (for fixed p), and maximizing $\min_{i \in M} \ell_i$, among others. To solve this problem, we can round the instance and state an IP analogous to [conf-IP] but considering an objective function. However, the objective function prevents us to use the main theorem as it is stated. To get over this issue, we study several ILPs. In each ILP we consider x_c to be a variable only if c has a given load, and fix the rest to be some optimal solution. Applying to each such ILP Theorem 1, plus some extra ideas, yields an analogous structural theorem. Afterwards, an algorithm similar to the one for makespan minimization yields the desired PTAS.

From an structural point of view, our sparsification lemma has other consequences on the structure of the knapsack polytope and the LP-relaxation of the [conf-IP]. More precisely, we can show that any vertex of the convex hull of Q must be simple. This, for example, helps us to upper bound the number of vertices by $2^{O(\log^2(T)+\log^2(d))}$. Moreover, we can show that the configuration-LP, obtained by replacing the integrality restriction in [conf-IP] by $x \geq 0$, if it is feasible then admits a solution whose support consist purely of simple configurations. Due to space limitations we leave many details and proofs to the full version.

2 Preliminaries

We will use the following notation throughout the paper. By default $\log(\cdot) = \log_2(\cdot)$, unless stated otherwise. Given two sets A, I, we will denote by A^I the set of all vectors indexed by I with entries in A, that is, $A^I = \{(a_i)_{i \in I} : a_i \in A \text{ for all } i \in I\}$. Moreover, for $A \subseteq \mathbb{R}$, we denote the support of a vector $a \in A^I$ as $\text{supp}(a) = \{i \in I : a_i \neq 0\}$.

We consider an arbitrary knapsack polytope $\mathcal{P} = \{c \in \mathbb{R}^d_{\geq 0} : \pi \cdot c \leq T\}$ where $\pi \in \mathbb{Z}^d_{>0}$ is a non-negative integral (row) vector and T is a positive integer. We assume without loss of generality that each coordinate π_k of π is upper bounded by T (otherwise $c_k = 0$ for all $c \in \mathbb{Z}^d \cap \mathcal{P}$). We focus on the set of integral vectors in $\mathcal{P}$ which we denote by $Q = \mathbb{Z}^d \cap \mathcal{P}$. We call an element $c \in Q$ a *configuration*. Given $b \in \mathbb{R}^d$, consider the problem of decomposing b as a conic integral combination of m configurations. That is, our aim is to find a feasible solution to [conf-IP], defined above.

A crucial property of the [conf-IP] is that there is always a solution with a support of small cardinality. This follows from a Caratheodory-type bound obtained by Eisenbrand and Shmonin [4]. Since we will need the argument later, we state the result applied to our case and revise its (very elegant) proof. We split the proof in two lemmas.

For a given subset $A \subseteq Q$, let us denote by x^A the indicator vector of A, that is $x^A_c = 1$ if $c \in A$, and 0 otherwise. Let us also denote by M the $(d+1) \times |Q|$ matrix that defines the system of equalities (1) and (2).

▶ **Lemma 2** (Eisenbrand and Shmonin [4]). *Let $x \in \mathbb{Z}^Q_{\geq 0}$ be a vector such that $|\operatorname{supp}(x)| > 2(d+1)\log(4(d+1)T)$. Then there exist two disjoint sets A, B with $\emptyset \neq A, B \subseteq \operatorname{supp}(x)$ such that $Mx^A = Mx^B$.*

▶ **Lemma 3** (Eisenbrand and Shmonin [4]). *If [conf-IP] is feasible, then there exists a feasible solution x such that $|\operatorname{supp}(x)| \leq 2(d+1)\log(4(d+1)T)$.*

Proof. Let x be a solution to [conf-IP] that minimizes $|\operatorname{supp}(x)| = s$. Assume by contradiction that $s > 2(d+1)\log(4(d+1)T)$. We show that we can find another solution x' to [conf-IP] with $|\operatorname{supp}(x')| < |\operatorname{supp}(x)|$, contradicting the minimality of $|\operatorname{supp}(x)|$. By Lemma 2, there exist two disjoint subsets $A, B \in \operatorname{supp}(x)$ such that $Mx^A = Mx^B$. Moreover, let $\lambda = \min\{x_c : c \in A\}$. Vector $x' := x - \lambda x^A + \lambda x^B$ is also a solution to [conf-IP] and has a strictly smaller support since a configuration $c^* \in \arg\min\{x_c : c \in A\}$ satisfies $x'_{c^*} = 0$. ◀

3 Structural Results

Recall that we call a configuration c simple if $|\operatorname{supp}(c)| \leq \log(T+1)$ and complex otherwise. An important observation to show Theorem 1 is that if c is a complex configuration, then $2c$ can be written as the sum of two configurations of smaller support. This is shown by the following Sparsification Lemma.

▶ **Lemma 4** (Sparsification Lemma). *Let $c \in Q$ be a complex configuration. Then there exist two configurations $c_1, c_2 \in Q$ such that*

1. $\pi \cdot c_1 = \pi \cdot c_2 = \pi \cdot c$,
2. $2c = c_1 + c_2$,
3. $\operatorname{supp}(c_1) \subsetneq \operatorname{supp}(c)$ *and* $\operatorname{supp}(c_2) \subsetneq \operatorname{supp}(c)$.

Proof. Consider for each subset $S \subseteq \operatorname{supp}(c)$, a configuration $c^S \in Q$ such that $c^S_i = c_i$ if $i \in S$ and $c^S = 0$ otherwise. As the number of subsets of $\operatorname{supp}(c)$ is $2^{|\operatorname{supp}(c)|}$, and $c^R \neq c^S$ if and only if $R \neq S$, the collection of vectors $V := \{c^S : S \subseteq \operatorname{supp}(c)\}$ has cardinality $|V| = 2^{|\operatorname{supp}(c)|}$.

On the other hand, for any vector $c^S \in V$ it holds that $\pi \cdot c^S \leq \pi \cdot c \leq T$. Hence, $\pi \cdot c^S \in \{0, 1 \ldots, T\}$ can take only $T+1$ different values. Using that c is a complex configuration and hence $2^{|\operatorname{supp}(c)|} > 2^{\log(T+1)} = T+1$, the pigeonhole principle ensures that there are two different non-empty configurations $c^S, c^R \subseteq V$ with $\pi \cdot c^S = \pi \cdot c^R$. By removing the intersection, we can assume w.l.o.g. that S and R have no intersection. We define $c_1 = c - c^S + c^R$ and $c_2 = c - c^R + c^S$, which satisfy the properties of the lemma as

$$\pi \cdot c_1 = \pi \cdot c - \pi \cdot c^S + \pi \cdot c^R = \pi \cdot c \quad \text{and}$$
$$2c = c - c^S + c^R + c - c^R + c^S = c_1 + c_2.$$

Since $\operatorname{supp}(c_1) \subseteq \operatorname{supp}(c) \setminus S$ and $\operatorname{supp}(c_2) \subseteq \operatorname{supp}(c) \setminus R$, property 3 is satisfied. ◀

With Lemma 4 we are ready to show Theorem 1. For the proof it is tempting to apply the lemma iteratively, replacing any complex configuration that is used twice by two configurations with smaller support. This can be repeated until there is no complex configuration taken multiple times. Then we can apply the technique of Lemma 3 to the obtained solution to bound the cardinality of the support. However, the last step might break the structure obtained if the solution implied by Lemma 3 uses a complex configuration more than once. In order to avoid this issue we consider a potential function. We show that a vector minimizing the chosen potential uses each complex configuration at most once, and that the number of complex configurations in the support is bounded. Finally, we apply the techniques from Lemma 3 restricted to variables corresponding to simple configurations.

Proof of Theorem 1. Consider the following potential function of a solution $x \in \mathbb{Z}^Q_{\geq 0}$ of [conf-IP],

$$\Phi(x) = \sum_{\text{complex config. } c} x_c |\operatorname{supp}(c)|.$$

Let x be a solution of [conf-IP] with minimum potential $\Phi(x)$, which is well defined since the set of feasible solutions has finite cardinality. We show two properties of x.

P1: *$x_c \leq 1$ for each complex configuration $c \in Q$.*

Assume otherwise. Consider the two configurations c_1 and c_2 implied by the previous lemma. We define a new solution $x'_e = x_e$ for $e \notin \{c, c_1, c_2\}$, $x'_{c_1} = x_{c_1} + 1$, $x'_{c_2} = x_{c_2} + 1$ and $x'_c = x_c - 2$. Since $|\operatorname{supp}(c_1)| < |\operatorname{supp}(c)|$ and $|\operatorname{supp}(c_2)| < |\operatorname{supp}(c)|$, we obtain that $\Phi(x') < \Phi(x)$ which contradicts the minimality of $\Phi(x)$.

P2: *The number of complex configurations in* $\operatorname{supp}(x)$ *is at most* $2(d+1)\log(4(d+1)T)$.

Let $\tilde{x}$ be the vector defined as $\tilde{x}_c = x_c$ if $c \in Q$ is complex, and $\tilde{x} = 0$ if $c \in Q$ is simple. Then Lemma 2 implies that there are exist two disjoint subsets $A, B \subseteq \operatorname{supp}(\tilde{x})$ of complex configurations such that $Mx^A = Mx^B$. Thus, the solution $x' = x - x^A + x^B$ and the solution $x'' = x - x^B + x^A$ are feasible for [config-IP]. By linearity, the potential function on the new solutions are $\Phi(x') = \Phi(x) - \Phi(x^A) + \Phi(x^B)$ or respectively $\Phi(x'') = \Phi(x) - \Phi(x^B) + \Phi(x^A)$. If $\Phi(x^A) > \Phi(x^B)$ or $\Phi(x^B) > \Phi(x^A)$ then we have constructed a new solution with smaller potential, contradicting our assumption on the minimality of $\Phi(x)$. We conclude that $\Phi(x^B) = \Phi(x^A)$ and thus $\Phi(x) = \Phi(x')$. By construction of x', we obtain that $x'_c > x_c \geq 1$ for any complex configuration $c \in B$. Having multiplicity ≥ 2 for a complex configuration c, we can proceed as in Case 1 to find a new solution with decreased potential, which yields a contradiction.

Given these two properties, to conclude the theorem it suffices to upper bound the number of simple configurations by $2(d+1)\log(4(d+1)T)$. Suppose this property is violated, then we find two sets $A, B \subseteq \operatorname{supp}(x)$ of simple configurations (see Lemma 2) with $Mx^A = Mx^B$ and proceed as in Lemma 3. Since Lemma 3 is only applied to simple configurations, properties P1 and P2 continue to hold and the theorem follows. ◀

Our techniques, in particular our Sparsification Lemma, imply two corollaries on the structure of the knapsack polytope and the LP-relaxation implied by the [conf-IP].

▶ **Corollary 5.** *Every vertex of conv.hull(Q) is a simple configuration. Moreover, the total number of simple configurations in Q is upper bounded by $2^{O(\log^2(T)+\log^2(d))}$ and thus the same expression upper bounds the number of vertices of conv.hull(Q).*

The following corollary follows as each complex configuration can be represented by a convex combination of simple configurations.

▶ **Corollary 6.** *Let [conf-LP] be the LP relaxation of [conf-IP], obtained by changing each constraint $x_c \in \mathbb{Z}_{\geq 0}$ to $x_c \geq 0$ for all $c \in Q$. If the LP is feasible then there exists a solution x such that each configuration $c \in \operatorname{supp}(x)$ is simple.*

4 Applications to Scheduling on Parallel Machines

In what follows we show how to exploit the structural insights of the previous section to derive faster algorithms for parallel machines scheduling problems. We start by considering $P||C_{\max}$, where we seek to assign a set of jobs $\mathcal{J}$ with processing times $p_j \in \mathbb{Z}_{>0}$ to a set $\mathcal{M}$ of m machines. For a given assignment $a : \mathcal{J} \mapsto \mathcal{M}$, we define the load of a machine i as $\sum_{j:a(j)=i} p_j$ and the *makespan* as the maximum load of jobs over all machines, which is the minimum time needed to complete the execution of all jobs on the processors. The goal is to find an assignment $\mathcal{J} \mapsto \mathcal{M}$ that minimizes the makespan.

We first follow well known rounding techniques [1, 2, 9, 8]. Consider an error tolerance $0 < \varepsilon < 1/3$ such that $1/\varepsilon^2$ is an integer. To get an estimation of the optimal makespan, we follow the standard dual approximation approach. First, we can use, e.g., the 2-approximation algorithm by Graham [6] to get an initial guess of the optimal makespan. Using binary search, we can then estimate the optimal makespan within a factor of $(1+\varepsilon)$ in $O(\log(1/\varepsilon))$ iterations. Therefore, it remains to give an algorithm that decides for a given makespan T, if there exists an assignment with makespan $(1+O(\varepsilon))T$ or reports that there exists no assignment with makespan $\leq T$.

For a given makespan T we define the set of big jobs $\mathcal{J}_{\text{big}} = \{j \in \mathcal{J} : p_j \geq \varepsilon T\}$ and the set of small jobs $\mathcal{J}_{\text{small}} = \mathcal{J} \setminus \mathcal{J}_{\text{big}}$. The following lemma shows that small jobs can be replaced from the instance by adding big jobs, each of size εT, as placeholders. Let S be the sum of processing times of jobs in $\mathcal{J}_{\text{small}}$ and let S^* denote the next value of S rounded up to the next multiple of εT, that is, $S^* = \varepsilon T \cdot \lceil S/(\varepsilon T) \rceil$. We define a new instance containing only big jobs by $\mathcal{J}^* = \mathcal{J}_{\text{big}} \cup \mathcal{J}_{\text{new}}$, where $\mathcal{J}_{\text{new}}$ contains $S^*/(\varepsilon T) \in \mathbb{N}$ jobs of size εT. The proof of the next lemma and the rest of the missing proofs of this section can be found in the full version.

▶ **Lemma 7.** *Given a feasible assignment $a : \mathcal{J} \mapsto \mathcal{M}$ of jobs with makespan T. Then there exists a feasible assignment $a_B : \mathcal{J}^* \mapsto \mathcal{M}$ of makespan $T^* \leq (1+\varepsilon)T$. Similarly, an assignment of jobs in $\mathcal{J}^*$ of makespan T^* can be transformed to an assignment of $\mathcal{J}$ of makespan at most $(1+\varepsilon)T^*$.*

By scaling the processing times of jobs in $\mathcal{J}^*$, we can assume that the makespan T has value $1/\varepsilon^2$. Also notice that we can assume that $p_j \leq T$ for all j, otherwise we cannot pack all jobs within makespan T. This implies that each job $j \in \mathcal{J}^*$ has a processing time of $1/\varepsilon \leq p_j \leq 1/\varepsilon^2$. In the following we give a transformation of big jobs in $\mathcal{J}^*$ by rounding their processing times. We first round the jobs to the next power of $1+\varepsilon$ as $p'_j = (1+\varepsilon)^{\lceil \log_{(1+\varepsilon)} p_j \rceil}$, and thus all rounded processing times belong to $\Pi' = \{(1+\varepsilon)^k : 1/\varepsilon \leq (1+\varepsilon)^k \leq (1+\varepsilon)/\varepsilon^2 \text{ and } k \in \mathbb{N}\}$. We further round processing times p'_j to the next integer $\bar{p}_j = \lceil p'_j \rceil$ and define a new set $\Pi = \{\lceil p \rceil : p \in \Pi'\}$. Notice that Π only contains integers and $|\Pi| \leq |\Pi'| \in O((1/\varepsilon)\log(1/\varepsilon))$.

▶ **Lemma 8.** *If there is a feasible schedule of jobs $\mathcal{J}^*$ with processing times p_j onto m machines with makespan $T^* \leq (1+\varepsilon)T$, then there is also a feasible schedule of jobs $\mathcal{J}^*$ with*

rounded processing $\bar{p}_j$ with a makespan of at most $(1+5\varepsilon)T$. Furthermore, the number of different processing times is at most $|\Pi| \in O((1/\varepsilon)\log(1/\varepsilon))$.

In what follows we give an algorithm that decides in polynomial time the existence of a solution for instance $\mathcal{J}^*$ with processing times $\bar{p}_j$ and makespan $\bar{T} = \lfloor (1+5\varepsilon)T \rfloor$. We call numbers in Π by $\pi_1, \ldots, \pi_d$ and define the vector $\pi = (\pi_1, \pi_2, \ldots, \pi_d) \in \mathbb{N}^d$ of rounded processing times. We consider *configurations* to be vectors in $Q = \mathcal{P} \cap \mathbb{Z}^d$, where $\mathcal{P} = \{c \in \mathbb{R}^d_{\geq 0} : \pi \cdot c \leq \bar{T}\}$ is a knapsack polytope (see Section 3). As before, we say that a configuration is simple if $|\operatorname{supp}(c)| \leq \log(\bar{T}+1)$, and complex otherwise. For a given assignment of jobs to machines, we say that a machine follows a configuration c if c_k is the number of jobs of size π_k assigned to the machine. We denote by $Q_c \subseteq Q$ the set of complex configurations and by $Q_s \subseteq Q$ the set of simple configurations.

Let b_k be the number of jobs of size π_k in the instance $\mathcal{J}^*$ (with processing times $\bar{p}$). Consider an ILP with integer variables x_c for each $c \in Q$, which denote the number of machines that follow configuration c. With these parameters the problem of scheduling all jobs in a solution of makespan $\bar{T}$ is equivalent to finding a solution to [conf-IP]. To solve the ILP we use, among other techniques, Kannan's algorithm [12] which is an improvement on the algorithm by Lenstra [13]. The algorithm has a running time of $2^{O(N \log N)} s$ where N is the number of variables and s is number of bits used to encode the input of the ILP in binary.

By Theorem 1, if [conf-IP] is feasible then there exists a thin solution. In particular if one configuration c is used by more than one machine then c is simple, and the total number of used configurations is $4(d+1)\log(4(d+1)\bar{T}) \in O((1/\varepsilon)\log^2(1/\varepsilon))$. Additionally, the number of machines following a complex configurations is at most $2(d+1)\log(4(d+1)\bar{T}) \in O((1/\varepsilon)\log^2(1/\varepsilon))$. We consider the following strategy to decide the existence of a schedule of makespan $\bar{T}$.

▶ Algorithm 9.

1. For each processing time π_k, guess the number $b^c_k \leq b_k$ of jobs covered by complex configurations.
2. Find a minimum number of machines m^c to schedule jobs b^c with makespan $\bar{T}$.
3. Guess the support of simple configurations $\bar{Q}_s \subseteq Q_s$ used by a thin solution, with $|\bar{Q}_s| \leq 4(d+1)\log(4(d+1)\bar{T}) \in O((1/\varepsilon)\log^2(1/\varepsilon))$.
4. Solve the ILP restricted to configurations in $\bar{Q}_s$:

$$\sum_{c \in \bar{Q}_s} c \cdot x_c = b - b^c,$$
$$\sum_{c \in \bar{Q}_s} x_c = m - m^c,$$
$$x_c \in \mathbb{Z}_{\geq 0} \qquad \text{for all } c \in \bar{Q}_s.$$

One of the key observations to prove the running time of the algorithm is that the number of simple configurations $|Q_s|$ is bounded by a quasi polynomial term:

$$|Q_s| \leq 2^{O(\log^2(1/\varepsilon))}.$$

This follows easily by Corollary 5, using that $|\bar{T}| \in O(1/\varepsilon^2)$ and $d = |\Pi| \in O((1/\varepsilon)\log(1/\varepsilon))$.

▶ **Lemma 10.** *Algorithm 9 can be implemented with a running time of $2^{O((1/\varepsilon)\log^4(1/\varepsilon))}\log(n)$.*

Proof. In step 1, the algorithm guesses which jobs are processed on machines following a complex configurations. Since each configuration contains at most $O(1/\varepsilon)$ jobs, there are at most $O(m^c/\varepsilon) = O((1/\varepsilon^2)\log^2(1/\varepsilon))$ jobs assigned to such machines. For each size $\pi_k \in \Pi$, we guess the number b_k^c of jobs of size π_k assigned to such machines. Hence, we can enumerate all possibilities for jobs assigned to complex machines in time $2^{O((1/\varepsilon)\log^2(1/\varepsilon))}$. After guessing the jobs, we can assign them to a minimum number of machines in step 2 (with makespan $\bar{T}$) with a simple dynamic program that stores vectors $(\ell, z_1, \ldots, z_d)$ with $z_k \leq b_k^c$ being the number of jobs of size π_k used in the first $\ell \leq m^c$ processors [11]. The size of the dynamic programming table is $O(m^c \prod_{k=1}^{d}(b_k^c+1))$. For any vector $(\ell, z_1, \ldots, z_d)$, determining whether it corresponds to a feasible solution can be done by checking all vectors of the type $(\ell-1, z_1', \ldots, z_d')$ for $z_k' \leq z_k$. Thus, the running time of the dynamic program is $O(m^c[\prod_{k=1}^{d}(b_k^c+1)]^2)$. Since $b_k^c \in O((1/\varepsilon^2)\log^2(1/\varepsilon))$ for each k, recalling that $m^c \in O((1/\varepsilon)\log^2(1/\varepsilon))$, and that $d = |\Pi| \in O((1/\varepsilon)\log(1/\varepsilon))$, we obtain that step 2 can be implemented with $2^{O((1/\varepsilon)\log^2(1/\varepsilon))}$ running time.

In step 3, our algorithm guesses the support of a thin solution x. Recall that if x is thin then $|\operatorname{supp}(x)| \leq 4(d+1)\log(4(d+1)\bar{T}) = O((1/\varepsilon)\log^2(1/\varepsilon))$. Let $D = 4(d+1)\log(4(d+1)\bar{T})$. Then this guess can be done in time

$$\sum_{i=0}^{D} \binom{|Q_s|}{i} \leq (D+1)|Q_s|^D \leq 2^{O((1/\varepsilon)\log^4(1/\varepsilon))}.$$

We remark that for this step is that thin solutions are particularly useful. Indeed, guessing the support on the original ILP takes time $2^{O((1/\varepsilon)^2\log^3(1/\varepsilon))}$.

In step 4, the restricted ILP with $4(d+1)\log(4(d+1)\bar{T}) = O((1/\varepsilon)\log^2(1/\varepsilon))$ variables is solved. Moreover, the size of the input can be bounded by $O((1/\varepsilon^2)\log^3(1/\varepsilon)\log(n))$. Running Kannan's algorithm [12] to solve the ILP takes time $2^{O((1/\varepsilon)\log^3(1/\varepsilon))}\log(n)$. Hence, the total running time of our algorithm can be bounded by $2^{O((1/\varepsilon)\log^4(1/\varepsilon))}\log(n)$. ◀

Putting all pieces together, we conclude with the following theorem.

▶ **Theorem 11.** *The minimum makespan problem on parallel machines $P||C_{\max}$ admits an EPTAS with running time $2^{O((1/\varepsilon)\log^4(1/\varepsilon))} + O(n\log n)$.*

4.1 Extension to other objectives

We now consider a more general family of objective functions defined by Alon et al. [1, 2]. For a fixed function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, we consider the following two objective functions: (I) $\min \sum_{i\in\mathcal{M}} f(\ell_i)$, and (II) $\min \max_{i\in\mathcal{M}} f(\ell_i)$, where ℓ_i denotes the load of machine i. Analogously, we study maximization versions of the problems: (I') $\max \sum_{i\in\mathcal{M}} f(\ell_i)$ and (II') $\max \min_{i\in\mathcal{M}} f(\ell_i)$.

For the minimization versions of the problem we assume that f is convex, while for (I') and (II') we assume it is concave. Moreover, we will need that the function satisfies the following sensitivity condition.

▶ **Condition 12.** *For all $\varepsilon > 0$ there exists $\delta = \delta(\varepsilon) > 0$ such that for all $x, y \in \mathbb{R}_{\geq 0}$,*

$$(1-\delta)y \leq x \leq (1+\delta)y \quad \Rightarrow \quad (1-\varepsilon)f(y) \leq f(x) \leq (1+\varepsilon)f(y).$$

Alon et al. showed that each problem in that family admits a PTAS with running time $h(\varepsilon)+O(n\log n)$, where $h(\varepsilon)$ is a constant term that depends only on ε. Moreover, if $\delta(\varepsilon)$ in the condition further satisfies that $1/(\delta(\varepsilon)) \in O(1/\varepsilon)$, the running time is $2^{(1/\varepsilon)^{\mathrm{poly}(1/\varepsilon)}} + O(n\log n)$.

In what follows we show how to improve this dependency if we have the additional condition that function f is non-decreasing, i.e., for all $0 \leq x \leq y$ we have that $f(x) \leq f(y)$. Since $1/(\delta(\varepsilon)) \in O(1/\varepsilon)$, we know that, for small enough ε, there exists a constant γ (independent of ε and δ) such that $1/\delta \leq \gamma/\varepsilon$. Moreover, we can assume w.l.o.g. that $\delta \leq \varepsilon$, and thus $\delta \leq \varepsilon \leq \gamma\delta$.

It is worth noticing that many interesting functions belong to this family. In particular (II) with $f(x) = x$ corresponds to the minimum makespan problem, (I) with $f(x) = x^p$, for constant p, corresponds to a problem that is equivalent to minimizing the L_p-norm of the vector of loads. Similarly, (II') with $f(x) = x$ corresponds to maximizing the minimum machine load. Notice that for all those objectives we have that $1/\delta = O(1/\varepsilon)$.

The techniques of Alon et al. are based on a rounding method and then solving an ILP. We based our results in the same rounding techniques and extend them further. We show that to obtain a PTAS in time $2^{O((1/\varepsilon)\log^4(1/\varepsilon))} + O(n \log n)$ it suffices to obtain a $(1 + O(\varepsilon))$-approximate solution to the rounded instance in the same running time. The details of the rounding are given in the full version.

Let $L = \sum_j p_j/m$ be the average machine load (of the original instance). After our rounding we obtain an instance I' with job set $\mathcal{J}'$ and processing times $\bar{p}_j$ for $j \in \mathcal{J}'$. Moreover, the $\bar{p}_j$ are multiples of L/λ^2, where $\lambda \geq 1/\delta$ is an integer such that $\lambda = O(1/\delta)$, and also $\bar{p}_j \geq L/\lambda$. It holds that there exists an optimal solution of the rounded instance with makespan at most $4L$ (see full version). Let $\Pi = \{\pi_1, \ldots, \pi_d\}$ be the distinct values that the processing times $\bar{p}_j$ can take. Our rounding guarantees that $d = |\Pi| = O((1/\delta)\log(1/\delta))$. We consider the knapsack polytope with capacity $\bar{T} := 4L$, that is $\mathcal{P} = \{c \in \mathbb{R}^d_{\geq 0} : \pi \cdot c \leq \bar{T}\}$. Notice that π and $\bar{T}$ are integer multiples of L/λ^2, and that $\mathcal{P}$ can also be written as $\{c \in \mathbb{R}^d_{\geq 0} : \pi/(L/\lambda^2) \cdot c \leq \bar{T}/(L/\lambda^2)\}$.

As before, we say that a configuration is simple if $|\operatorname{supp}(c)| \leq \log(\bar{T} + 1)$, and complex otherwise. We denote by $Q_c \subseteq Q$ the set of complex configurations and by $Q_s \subseteq Q$ the set of simple configurations. In what follows we focus on objective function (I).

We set an ILP for the problem as before. Notice that each configuration c incurs a cost of $f_c := f(\pi \cdot c)$. Moreover, we round and scale the values f_c by defining $\bar{f}_c = \lceil f_c/(\varepsilon f_{\min})\rceil$, where $f_{\min} = \min_{c \in Q} f_c$. It is not hard to see that solving a problem with those coefficients yields a $(1 + \varepsilon)$-approximate solution to the optimal solution of I' with processing times $\bar{p}_j$. Let also b_k be the number of jobs j of processing time $\bar{p}_j = \pi_k$ in $\mathcal{J}'$. Consider the ILP obtained by adding to [conf-IP] the objective function $\min \sum_{c \in Q} \bar{f}_c \cdot x_c$. We call this ILP [cost-conf-IP]. With our previous discussion, it suffices to solve this ILP optimally. To solve this problem, we first notice that the largest coefficient in the objective can be bounded as follows.

▶ **Lemma 13.** *If f satisfies Condition 12 and it is non-decreasing, then the largest value $\max_{c \in Q} \bar{f}_c$ is upper bounded by $1/\delta^{O(1)}$.*

As we now must consider the objective function, we cannot simply apply Theorem 1 to [cost-conf-ILP]. However, we can prove a slightly weaker version by decomposing the ILP in several smaller ones and applying the theorem to each of them.

▶ **Theorem 14.** *If* [cost-conf-IP] *is feasible, then there exists an optimal solution x satisfying:*

1. $\sum_{c \in Q_c} x_c \in O((1/\delta^3)\log^2(1/\delta))$, *and*
2. $|\operatorname{supp}(x) \cap Q_s| \in O((1/\delta)\log^2(1/\delta))$.

Proof. Notice that the load of each configuration $\pi \cdot c$ is a multiple of L/λ^2, and thus $\pi \cdot c \in \{L/\lambda, L/\lambda + L/(\lambda^2), \ldots, 4L\}$. We classify the configurations according to their loads,

$Q^\ell := \{c \in Q : \pi \cdot c = L/\lambda + \ell \cdot L/(\lambda^2)\}$, for $\ell \in \{0, \dots, 4\lambda^2 - \lambda\}$. Let x^* be an optimal solution of [cost-conf-IP]. Then we can considered an ILP for each load value ℓ:

$$[\text{conf-IP}]_\ell \quad \sum_{c \in Q^\ell} c \cdot x_c = \sum_{c \in Q^\ell} c \cdot x^*_c, \tag{4}$$

$$\sum_{c \in Q^\ell} x_c = \sum_{c \in Q^\ell} x^*_c, \tag{5}$$

$$x_c \in \mathbb{Z}_{\geq 0} \qquad \text{for all } c \in Q^\ell. \tag{6}$$

Scaling π by multiplying it by λ^2/L we obtain an integral vector (since π is an integer multiple of $L/(\lambda^2)$), we can apply Theorem 1 to each ILP $[\text{conf-IP}]_\ell$, which yields that there exists a thin solution x^ℓ. In particular the number of complex configurations in x^ℓ is $\sum_{c \in Q_c \cap Q^\ell} x^\ell_c \in O((1/\delta)\log^2(1/\delta))$. Since $\bar{f}_c$ depends only on the load of c, concatenating these solutions yields a solution $x' := (x^\ell)_\ell$ that is optimal for [cost-conf-IP], such that $\sum_{c \in Q_c} x'_c \in O((\lambda^2) \cdot (1/\delta)\log^2(1/\delta)) = O((1/\delta^3)\log^2(1/\delta))$. It remains to bound the number of simple configurations in the support. To this end, we consider the ILP restricted to simple configurations as follows:

$$[\text{cost-conf-IP}]_s \quad \min \sum_{c \in Q_s} \bar{f}_c \cdot x_c$$

$$\sum_{c \in Q_s} c \cdot x_c = b - \sum_{c \in Q_c} c \cdot x'_c, \tag{7}$$

$$\sum_{c \in Q_s} x_c = m - \sum_{c \in Q_c} x'_c, \tag{8}$$

$$x_c \in \mathbb{Z}_{\geq 0} \qquad \text{for all } c \in Q_s. \tag{9}$$

We apply the result of Eisenbrand and Shmonin [4] to this ILP. In its more general form, this result ensures the existence of a solution x'' with support of size $O(N(\log(N)+\Delta))$, where N is the number of restrictions and Δ is the encoding size of the largest coefficient appearing in the cost vector and restriction matrix. In our case $N = d + 1 = O((1/\delta)\log(1/\delta))$, and $\Delta = O(\log(\max\{1/\delta, \max_{c \in Q} \bar{f}_c)\}) = O(\log(1/\delta))$ (Lemma 13). Thus $O(N(\log(N) + \Delta)) = O((1/\delta)\log^2(1/\delta))$. The theorem follows by concatenating $(x''_c)_{c \in Q_s}$ with $(x'_c)_{c \in Q_c}$. ◀

Finally, we use the structure given by the theorem to solve this ILP optimally. The idea is similar to Algorithm 9 and thus we defer the details to the full version.

▶ **Theorem 15.** *Consider the scheduling problem on parallel machines with objective functions* (I), (II) *for f convex (respectively* (I') *and* (II') *for f concave). If f satisfies Condition 12 for $1/\delta = O(1/\varepsilon)$ and it is non-decreasing, then the problem admits an EPTAS with running time $2^{O((1/\varepsilon)\log^4(1/\varepsilon))} + O(n \log n)$.*

5 Minimum makespan scheduling on uniform machines

In this section we generalize our result for $P||C_{\max}$ to uniform machines. Consider a set of jobs $\mathcal{J}$ with processing times p_j and a set of m non-identical machines $\mathcal{M}$ where machine $i \in \mathcal{M}$ runs at speed s_i. If job j is executed on machine i the machine needs p_j/s_i time units to complete the job. The problem is to find an assignment $a : \mathcal{J} \to \mathcal{M}$ for the jobs to the machines that minimizes the makespan; $\max_i \sum_{j:a(j)=i} p_j/s_i$. The problem is denoted by $Q||C_{max}$. We suppose that $s_1 \geq s_2 \geq \ldots \geq s_m$. Jansen [10] found an efficient polynomial

time approximation scheme (EPTAS) for this scheduling problem which has a running time of $2^{O(1/\varepsilon^2 \log^3(1/\varepsilon))} + \text{poly}(n)$. Here we show how to improve the running time and prove the main result of this section.

▶ **Theorem 16.** *There is an EPTAS (a family of algorithms $\{A_\varepsilon : \varepsilon > 0\}$) which, given an instance I of $Q||C_{max}$ with n jobs and m machines and a positive number $\varepsilon > 0$, produces a schedule of makespan $A_\varepsilon(I) \leq (1+\varepsilon)\textsc{Opt}(I)$. The running time of A_ε is $2^{O(1/\varepsilon \log^4(1/\varepsilon))} + \text{poly}(n)$.*

Let $0 < \delta < \varepsilon$. We follow the approach by Jansen [10], transforming the scheduling problem into a bin packing problem with different bin capacities, rounding the processing times and bin capacities, and dividing the bins into at most three groups $\mathcal{B}_1, \mathcal{B}_2$ and $\mathcal{B}_3$ depending on the bin capacities.

Here, we focus on a special case which contains the main difficulty of the problem. The full exposition can be found in the full version. In group $\mathcal{B}_2$, the bins can have a capacity of value $\bar{c}(1) > \ldots > \bar{c}(L)$ for some $L \in O(1/\delta \log(1/\delta))$. We call B_ℓ the set of bins in $\mathcal{B}_2$ with capacity $\bar{c}(\ell)$. Our rounded instance contains jobs of sizes $\pi_1, \ldots, \pi_d$ for $d \in O(1/\delta \log(1/\delta))$. For each B_ℓ we consider configurations $\bar{C}_1^{(\ell)}, \ldots, \bar{C}_{\bar{h}_\ell}^{(\ell)}$. The configurations are defined using job of size at least $\delta\bar{c}(\ell)$ which are rounded up to multiples of $\delta^2\bar{c}(\ell)$. Thus, regarding these configurations (and only these configurations), jobs have sizes of the form $q(k,\ell)\delta^2\bar{c}(\ell)$ with $q(k,\ell) \in \mathbb{Z}^+$ and $k \in \{1,\ldots,d\}$. We denote by $a(k, \bar{C}_i^{(\ell)})$ the multiplicity of jobs of size $q(k,\ell)\delta^2\bar{c}(\ell)$ in configuration $\bar{C}_i^{(\ell)}$. The rounding implies also that the rounded size $size(\bar{C}_i^{(\ell)})$ of a configuration is a multiple of $\delta^2\bar{c}(\ell)$. Each such configuration corresponds to an integral point inside the knapsack polytope $\mathcal{P}_\ell = \{C = (a(k,C))_k : \sum_k q(k,\ell)\delta^2\bar{c}(\ell)a(k,C) \leq (1+\delta)\bar{c}(\ell)\}$.

Let $\bar{m}_\ell = |B_\ell|$ be the number of machines of capacity $\bar{c}(\ell)$. Consider a given solution to our scheduling problem. For this case we say that a bin in B_ℓ follows a configuration $\bar{C}_i^{(\ell)}$ if it has $a(k, \bar{C}_i^{(\ell)})$ jobs whose size, rounded to the next multiple of $\delta^2\bar{c}(\ell)$, equals to $q(k,\ell)\delta^2\bar{c}(\ell)$. Let $\bar{x}_i^{(\ell)}$ be the number of bins in B_ℓ that follows configuration $\bar{C}_i^{(\ell)}$. Notice that the configurations do not consider jobs of size smaller than $\delta\bar{c}(\ell)$ that might be assigned to a bin in B_ℓ. Consider the following ILP, which we denote by $[\text{conf-IP}]_Q$:

$$\sum_i x_i^{(\ell)} = \bar{m}_\ell \qquad \text{for } \ell = 1,\ldots,L,$$

$$\sum_{\ell,i} a(k, \bar{C}_i^{(\ell)}) x_i^{(\ell)} = \sum_{\ell,i} a(k, \bar{C}_i^{(\ell)}) \bar{x}_i^{(\ell)} \qquad \text{for } k = 1\ldots,d,$$

$$\sum_i \frac{size(\bar{C}_i^{(\ell)})}{\delta^2\bar{c}(\ell)} x_i^{(\ell)} = \sum_i \frac{size(\bar{C}_i^{(\ell)})}{\delta^2\bar{c}(\ell)} \bar{x}_i^{(\ell)},$$

$$x_i^{(\ell)} \geq 0 \,\text{integral} \qquad \text{for } i = 1,\ldots,\bar{h}_\ell, \ell = 1,\ldots,L.$$

The second equality of the ILP ensures that the solution constructed maintains the same subset of jobs larger than $\delta\bar{c}(\ell)$ to bins in B_ℓ as solution $\bar{x}$. The third equality ensures that we are leaving enough space for jobs not covered by a configuration (i.e., a job that is assigned within B_ℓ but whose size is less than $\delta\bar{c}(\ell)$). As in previous sections, a configuration $\bar{C}_i^{(\ell)}$ is called simple if $|\operatorname{supp}(\bar{C}_i^{(\ell)})| \leq \log(\bar{c}(\ell)(1+\delta)/(\delta^2\bar{c}(\ell)) + 1) = \log(1/\delta^2 + 1/\delta + 1)$ (here we are scaling the capacity of a configuration by $\delta^2\bar{c}(\ell)$, since all rounded job sizes are multiples of $\delta^2\bar{c}(\ell)$). Otherwise, we call a configuration $\bar{C}_i^{(\ell)}$ complex. Crucially, the ILP above satisfies that all coefficients are at most $\text{poly}(1/\delta)$. This allows us to generalize our result in Theorem 1 to our ILP above, with a similar proof technique as Theorem 14, which yields the following lemma. Using this lemma, we can define an algorithm similar to

the algorithm for the case of identical machines, to obtain an improved running time for $Q||C_{max}$ and prove Theorem 16.

▶ **Lemma 17.** *Assume that the ILP [conf-IP]$_Q$ is feasible and let S denote the set of all simple configurations. Then there exists a feasible solution x' such that:* (1) *If $x'^{(\ell)}_i > 1$ then the configuration $\bar{C}^{(\ell)}_i$ is simple,* (2) *the support of x' satisfies $|\operatorname{supp}(x') \cap S| \in O(1/\delta \log^2(1/\delta))$, and* (3) *the support of x' satisfies $|\operatorname{supp}(x') \setminus S| \in O(1/\delta^2 \log^3(1/\delta))$.*

References

1. N. Alon, Y. Azar, G. Woeginger, and T. Yadid. Approximation schemes for scheduling. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '97)*, pages 493–500. ACM/SIAM, 1997.
2. N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1:55–66, 1998.
3. L. Chen, K. Jansen, and G. Zhang. On the optimality of approximation schemes for the classical scheduling problem. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'13)*, pages 657–668. ACM/SIAM, 2013.
4. F. Eisenbrand and G. Shmonin. Carathéodory bounds for integer cones. *Operations Research Letters*, 34:564–568, 2006.
5. M. X. Goemans and T. Rothvoß. Polynomiality for bin packing with a constant number of item types. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*, pages 830–839. ACM/SIAM, 2014.
6. R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.
7. R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17:416–429, 1969.
8. D. Hochbaum, editor. *Approximation algorithms for NP-hard problems.* PWS Publishing Company, 1997.
9. D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM*, 34:144–162, 1987.
10. K. Jansen. An EPTAS for scheduling jobs on uniform processors: Using an MILP relaxation with a constant number of integral variables. *SIAM Journal on Discrete Mathematics*, 24:457–485, 2010.
11. K. Jansen and C. Robenek. Scheduling jobs on identical and uniform processors revisited. In *Approximation and Online Algorithms (WAOA'11)*, number 7164 in Lecture Notes in Computer Science, pages 109–122. Springer, 2011.
12. R. Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12:415–440, 1987.
13. H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.
14. J. Y-T. Leung. Bin packing with restricted piece sizes. *Information Processing Letters*, 31:145–149, 1989.