An EPTAS for Scheduling on Unrelated Machines of Few Different Types[*]

Klaus Jansen   Marten Maack

Department of Computer Science, University of Kiel, 24118 Kiel, Germany
{kj, mmaa}@informatik.uni-kiel.de

December 7, 2017

Abstract

In the classical problem of scheduling on unrelated parallel machines, a set of jobs has to be assigned to a set of machines. The jobs have a processing time depending on the machine and the goal is to minimize the makespan, that is the maximum machine load. It is well known that this problem is NP-hard and does not allow polynomial time approximation algorithms with approximation guarantees smaller than 1.5 unless P=NP. We consider the case that there are only a constant number K of machine types. Two machines have the same type if all jobs have the same processing time for them. This variant of the problem is strongly NP-hard already for $K = 1$. We present an efficient polynomial time approximation scheme (EPTAS) for the problem, that is, for any $\varepsilon > 0$ an assignment with makespan of length at most $(1 + \varepsilon)$ times the optimum can be found in polynomial time in the input length and the exponent is independent of $1/\varepsilon$. In particular we achieve a running time of $2^{\mathcal{O}(K \log(K)1/\varepsilon \log^4 1/\varepsilon)} + \mathrm{poly}(|I|)$, where $|I|$ denotes the input length. Furthermore, we study three other problem variants and present an EPTAS for each of them: The Santa Claus problem, where the minimum machine load has to be maximized; the case of scheduling on unrelated parallel machines with a constant number of uniform types, where machines of the same type behave like uniformly related machines; and the multidimensional vector scheduling variant of the problem where both the dimension and the number of machine types are constant. For the Santa Claus problem we achieve the same running time. The results are achieved, using mixed integer linear programming and rounding techniques.

1

1 Introduction

We consider the problem of scheduling jobs on unrelated parallel machines—or unrelated scheduling for short—in which a set $\mathcal{J}$ of n jobs has to be assigned to a set $\mathcal{M}$ of m machines. Each job j has a processing time p_{ij} for each machine i and the goal is to find a schedule $\sigma : \mathcal{J} \to \mathcal{M}$ minimizing the *makespan* $C_{\max}(\sigma) = \max_{i \in \mathcal{M}} \sum_{j \in \sigma^{-1}(i)} p_{ij}$, i.e. the maximum machine load. The problem is one of the classical scheduling problems studied in approximation. In 1990 Lenstra, Shmoys and Tardos [23] showed that there is no approximation algorithm with an approximation guarantee smaller than 1.5, unless P=NP. Moreover, they presented a 2-approximation, and closing this gap is a rather famous open problem in scheduling theory and approximation (see e.g. [27]).

In particular, we study the special case where there is only a constant number K of *machine types*. Two machines i and i' have the same type, if $p_{ij} = p_{i'j}$ holds for each job j. In many application scenarios this variant is plausible, e.g., when considering computers which typically only have a very limited number of different types of processing units. We denote the processing time of a job j on a machine of type $t \in [K]$ by p_{tj} and assume that the input consist of the corresponding $K \times n$ processing time matrix together with machine multiplicities m_t for each type t, yielding $m = \sum_{t \in [K]} m_t$. Note that the case $K = 1$ is equivalent to the classical scheduling on identical machines. We also study three other variants of the problem:

Santa Claus Problem. We consider the reverse objective of maximizing the minimum machine load, i.e. $C_{\min}(\sigma) = \min_{i \in \mathcal{M}} \sum_{j \in \sigma^{-1}(i)} p_{ij}$. This problem is known as max-min fair allocation or the Santa Claus problem. The intuition behind these names is that the jobs are interpreted as goods (e.g. presents), the machines as players (e.g. children), and the processing times as the values of the goods from the perspective of the different players. Finding an assignment that maximizes the minimum machine load, means therefore finding an allocation of the goods that is in some sense fair (making the least happy kid as happy as possible). We will refer to the problem as Santa Claus problem in the following, but otherwise will stick to the scheduling terminology.

Uniform Types. Two machines i and i' have the same *uniform* machine type, if there is a scaling factor s such that $p_{ij} = sp_{i'j}$ for each job j. While jobs behave on machines of the same type like they do on identical machines, they behave of machines of the same uniform type like they do on uniformly related machines. Hence, we may assume that the input consists of job sizes p_{tj} depending on the job j and the uniform type t, together with uniform machine types t_i and machine speeds s_i, such that $p_{ij} = p_{t_ij}/s_i$.

Vector Scheduling. In the D-dimensional vector scheduling variant of unrelated scheduling, a processing time vector $p_{ij} = (p_{ij}^{(1)}, \ldots, p_{ij}^{(D)})$ is given for each job j and machine i and the makespan of a schedule σ is defined as the maximum load any machine receives in any dimension:

$$C_{\max}(\sigma) = \max_{i \in \mathcal{M}} \Big\| \sum_{j \in \sigma^{-1}(i)} p_{ij} \Big\|_\infty = \max_{i \in \mathcal{M}, d \in [D]} \sum_{j \in \sigma^{-1}(i)} p_{ij}^{(d)}$$

Machine types are defined correspondingly. We consider the case that both K and D are constant and like in the one dimensional case we may assume that the input consist of processing time vectors depending on types and jobs, together with machine multiplicities.

Basic Concepts. We study polynomial time approximation algorithms: Given an instance I of an optimization problem, an α-approximation A for this problem produces a solution in time $\mathrm{poly}(|I|)$, where $|I|$ denotes the input length. For the objective function value $A(I)$ of this solution it is guaranteed that $A(I) \leq \alpha \mathrm{OPT}(I)$, in the case of an minimization problem, or $A(I) \geq (1/\alpha)\mathrm{OPT}(I)$, in the case of an maximization problem, where $\mathrm{OPT}(I)$ is the value of an optimal solution. We call α the *approximation guarantee* or *rate* of the algorithm. In some cases a polynomial time approximation scheme (PTAS) can be achieved, that is, an $(1+\varepsilon)$-approximation for each $\varepsilon > 0$. If for such a family of algorithms the running time can be bounded by $f(1/\varepsilon)\mathrm{poly}(|I|)$ for some computable function f, the PTAS is called *efficient* (EPTAS), and if the running time is polynomial in both $1/\varepsilon$ and $|I|$ it is called *fully polynomial* (FPTAS).

Related Work. It is well known that the unrelated scheduling problem admits an FPTAS in the case that the number of machines is considered constant [16] and we already mentioned the seminal work by Lenstra et al. [23]. Furthermore, the problem of unrelated scheduling with a constant number of machine types is strongly NP-hard, because it is a generalization of the strongly NP-hard problem of scheduling on identical parallel machines. Therefore an FPTAS can not be hoped for in this case. However, Wiese, Bonifaci and Baruah showed that there is a PTAS [26], and Wiese and Bonifaci [6] gave an extended analysis for the vector scheduling case where both the dimension D and K are constant. The authors do not present a detailed analysis of the running time, however the procedures involve guessing steps with $(m+1)^{K\kappa}$ possibilities, where $\kappa = (D/\varepsilon)^{\mathcal{O}((1/\varepsilon \log(D/\varepsilon))^D)}$. Gehrke, Jansen, Kraft and Schikowski [13] presented a PTAS with an improved running time of $\mathcal{O}(Kn) + m^{\mathcal{O}(K/\varepsilon^2)}(\log(m)/\varepsilon)^{\mathcal{O}(K^2)}$ for the regular one dimensional case of unrelated scheduling with a constant number of machine types. On the other

hand, Chen, Jansen and Zhang [9] showed that there is no PTAS for scheduling on identical machines with running time $2^{(1/\varepsilon)^{1-\delta}}$ for any $\delta > 0$, unless the exponential time hypothesis fails. Furthermore, the case $K = 2$ has been studied: Imreh [17] designed heuristic algorithms with rates $2 + (m_1 - 1)/m_2$ and $4 - 2/m_1$, and Bleuse et al. [5] presented an algorithm with rate $4/3 + 3/m_2$ and, moreover, a (faster) $3/2$-approximation, for the case that for each job the processing time on the second machine type is at most the one on the first. Moreover, Raravi and Nélis [25] designed a PTAS for the case with two machine types.

Interestingly, unrelated scheduling is in P, if both the number of machine types and the number of job types is bounded by a constant. This is implied by a recent result due to Chen, Marx, Ye and Zhang [10] building upon a result by Goemans and Rothvoss [14]. Job types are defined analogously to machine types, i.e., two jobs j, j' have the same type, if $p_{ij} = p_{ij'}$ for each machine i. In this case the matrix (p_{ij}) has only a constant number of distinct rows and columns. Note that both the number of machine types and uniform machine types bounds the rank of this matrix. However the case of unrelated scheduling where the matrix (p_{ij}) has constant rank turns out to be much harder: Already for the case with rank 3 the problem is APX-hard [10] and for rank 4 an approximation algorithm with rate smaller than $3/2$ can be ruled out, unless P=NP [11]. In a rather recent work, Knop and Koutecký [22] considered the number of machine types as a parameter from the perspective of fixed parameter tractability. They showed that unrelated scheduling is fixed parameter tractable for the parameters K and $\max p_{i,j}$, that is, there is an algorithm with running time $f(K, \max p_{i,j}) \text{poly}(|I|)$ for some computable function f that solves the problem to optimality. Chen et al. [10] extended this, showing that unrelated scheduling is fixed parameter tractable for the parameters $\max p_{i,j}$ and the rank of the processing time matrix.

For the case that the number of machines is constant, the Santa Claus problem behaves similar to the unrelated scheduling problem: there is an FPTAS that is implied by a result due to Woeginger [28]. In the general case however, so far no approximation algorithm with a constant approximation guarantee has been found. The results by Lenstra et al. [23] can be adapted to show that that there is no approximation algorithm with a rate smaller than 2, unless P=NP, and to get an algorithm that finds a solution with value at least $\text{OPT}(I) - \max p_{i,j}$, as was done by Bezáková and Dani [4]. Since $\max p_{i,j}$ could be bigger than $\text{OPT}(I)$, this does not provide a (multiplicative) approximation guarantee. Bezáková and Dani also presented a simple $(n - m + 1)$-approximation and an improved approximation guarantee of $\mathcal{O}(\sqrt{n} \log^3 n)$ was achieved by Asadpour and Saberi [2]. The best rate so far is $O(n^\varepsilon)$ due to Bateni et al. [3] and Chakrabarty et al. [7], with a running time of $\mathcal{O}(n^{1/\varepsilon})$ for any $\varepsilon > 0$.

To the best of our knowledge, unrelated scheduling with a constant number of uniform machine types has not been studied before, but we argue that it is a natural extension of the case with a constant number of regular machine types and also a sensible special case of the general unrelated scheduling and the low rank case in particular.

The vector scheduling problem has been studied for the special case of identical machines by Chekuri and Khanna [8]. They achieve a PTAS for the case that D is constant and an $\mathcal{O}(\log^2 D)$-approximation for the case that D is arbitrary.

Results and Methodology. The main result of this paper is the following:

Theorem 1. *There is an EPTAS for both scheduling on unrelated parallel machines and the Santa Claus problem with a constant number K of different machine types with running time $2^{\mathcal{O}(K \log(K)^{1/\varepsilon} \log^4 {1/\varepsilon})} + \mathrm{poly}(|I|)$.*

First we present a basic version of the EPTAS for unrelated scheduling with a running time doubly exponential in $1/\varepsilon$. For this EPTAS we use the dual approximation approach by Hochbaum and Shmoys [15] to get a guess T of the optimal makespan OPT. Then, we further simplify the problem via geometric rounding of the processing times. Next, we formulate a mixed integer linear program (MILP) with a constant number of integral variables that encodes a relaxed version of the problem. The MILP can be seen as a generalization of the classical integer linear program of configurations—or configuration ILP—for scheduling on identical parallel machines. We solve it with the algorithm by Lenstra and Kannan [24, 21]. The fractional variables of the MILP have to be rounded and we achieve this with a flow network utilizing flow integrality and causing only a small error. With an additional error the obtained solution can be used to construct a schedule with makespan $(1 + \mathcal{O}(\varepsilon))T$. This procedure is described in detail in Section 2. Building upon the basic EPTAS we achieve the improved running time using techniques by Jansen [18] and by Jansen, Klein and Verschae [19]. The basic idea of these techniques is to make use of existential results about simple structured solutions of integer linear programs (ILPs). In particular these results can be used to guess the non-zero variables of the MILP, because they sufficiently limit the search space. We show how these techniques can be applied in our case in Section 3. Furthermore, we present efficient approximation schemes for several other problem variants, thereby demonstrating the flexibility of our approach. In particular, we can adapt all our techniques to the Santa Claus problem yielding the result stated above. This is covered in Section 4 and in Section 5 we show:

Theorem 2. *There is an EPTAS for scheduling on unrelated parallel machines with a constant number K of different uniform machine types with running time $2^{\mathcal{O}(K \log(K)^{1/\varepsilon^3} \log^5 {1/\varepsilon})} + \mathrm{poly}(|I|)$.*

We achieve this with a non-trivial combination of the ideas of Section 2 with techniques for scheduling on uniformly related machines by Jansen [18]. Finally, in Section 6, we revisit the unrelated vector scheduling problem that was studied by Bonifaci and Wiese [6]. We show that an additional rounding step—similar to the one in [8]—together with a slight modification of the MILP and the rounding procedure yield an EPTAS for this problem as well.

Theorem 3. *There is an EPTAS for vector scheduling on unrelated parallel machines with constant dimension D a constant number K of different machine types.*

Note that our results may also be seen as fixed parameter tractable algorithms for the parameters $1/\varepsilon$ and K (and D). In the last section we elaborate on possible directions for future research.

2 Basic EPTAS

In this chapter we describe a basic EPTAS for unrelated scheduling with a constant number of machine types, with a running time doubly exponential in $1/\varepsilon$. Wlog. we assume $\varepsilon < 1$. Furthermore $\log(\cdot)$ denotes the logarithm with basis 2 and for $k \in \mathbb{Z}_{\geq 0}$ we write $[k]$ for $\{1, \ldots, k\}$.

First, we simplify the problem via the classical dual approximation concept by Hochbaum and Shmoys [15]. In the simplified version of the problem a target makespan T is given and the goal is to either output a schedule with makespan at most $(1 + \alpha\varepsilon)T$ for some constant $\alpha \in \mathbb{Z}_{>0}$, or correctly report that there is no schedule with makespan T. We can use a polynomial time algorithm for this problem in the design of a PTAS in the following way. First we obtain an upper bound B for the optimal makespan OPT of the instance with $B \leq 2\text{OPT}$. This can be done using the 2-approximation by Lenstra et al. [23]. With binary search on the interval $[B/2, B]$ we can find in $\mathcal{O}(\log 1/\varepsilon)$ iterations a value T^* for which the mentioned algorithm is successful, while $T^* - \varepsilon B/2$ is rejected. We have $T^* - \varepsilon B/2 \leq \text{OPT}$ and therefore $T^* \leq (1 + \varepsilon)\text{OPT}$. Hence the schedule we obtained for the target makespan T^* has makespan at most $(1 + \alpha\varepsilon)T^* \leq (1 + \alpha\varepsilon)(1 + \varepsilon)\text{OPT} = (1 + \mathcal{O}(\varepsilon))\text{OPT}$. In the following we will always assume that a target makespan T is given. Next we present a brief overview of the algorithm for the simplified problem followed by a more detailed description and analysis.

Algorithm 4.
  (i) Simplify the input via geometric rounding with an error of εT.
  (ii) Build the mixed integer linear program MILP($\bar{T}$) and solve it with the algorithm by Lenstra and Kannan ($\bar{T} = (1 + \varepsilon)T$).

(iii) If there is no solution, report that there is no solution with makespan T.

(iv) Generate an integral solution for $\mathrm{MILP}(\bar{T} + \varepsilon T + \varepsilon^2 T)$ via a flow network utilizing flow integrality.

(v) The integral solution is turned into a schedule with an additional error of $\varepsilon^2 T$ due to the small jobs.

Simplification of the Input. We construct a simplified instance $\bar{I}$ with modified processing times $\bar{p}_{tj}$. If a job j has a processing time bigger than T for a machine type $t \in [K]$ we set $\bar{p}_{tj} = \infty$. We call a job *big* (for machine type t), if $p_{tj} > \varepsilon^2 T$, and *small* otherwise. We perform a geometric rounding step for each job j with $p_{tj} < \infty$, that is we set $\bar{p}_{tj} = (1+\varepsilon)^x \varepsilon^2 T$ with $x = \lceil \log_{1+\varepsilon}(p_{tj}/(\varepsilon^2 T)) \rceil$.

Lemma 5. *If there is a schedule with makespan at most T for I, the same schedule has makespan at most $(1+\varepsilon)T$ for instance $\bar{I}$ and any schedule for instance $\bar{I}$ can be turned into a schedule for I without increase in the makespan.*

We will search for a schedule with makespan $\bar{T} = (1+\varepsilon)T$ for the rounded instance $\bar{I}$. We establish some notation for the rounded instance. For any rounded processing time p we denote the set of jobs j with $\bar{p}_{tj} = p$ by $J_t(p)$. Moreover, for each machine type t let S_t and B_t be the sets of small and big rounded processing times. Obviously we have $|S_t| + |B_t| \leq n$. Furthermore $|B_t|$ is bounded by a constant: Let N be such that $(1+\varepsilon)^N \varepsilon^2 T$ is the biggest rounded processing time for all machine type. Then we have $(1+\varepsilon)^{N-1} \varepsilon^2 T \leq T$ and therefore $|B_t| \leq N \leq \log(1/\varepsilon^2)/\log(1+\varepsilon) + 1 \leq 1/\varepsilon \log(1/\varepsilon^2) + 1$ (using $\varepsilon \leq 1$).

MILP. For any set of processing times P we call the P-indexed vectors of non-negative integers $\mathbb{Z}_{\geq 0}^P$ *configurations* (for P). The *size* $\mathrm{size}(C)$ of configuration C is given by $\sum_{p \in P} C_p p$. For each $t \in [K]$ we consider the set $\mathcal{C}_t(\bar{T})$ of configurations C for the big processing times B_t and with $\mathrm{size}(C) \leq \bar{T}$. Given a schedule σ, we say that a machine i of type t obeys a configuration C, if the number of big jobs with processing time p that σ assigns to i is exactly C_p for each $p \in B_t$. Since the processing times in B_t are bigger than $\varepsilon^2 T$ we have $\sum_{p \in B_t} C_p \leq 1/\varepsilon^2$ for each $C \in \mathcal{C}_t(\bar{T})$. Therefore the number of distinct configurations in $\mathcal{C}_t(\bar{T})$ can be bounded by $(1/\varepsilon^2 + 1)^N < (1/\varepsilon^2 + 1)^{1/\varepsilon \log(1/\varepsilon^2)+1} = 2^{\log(1/\varepsilon^2+1)1/\varepsilon \log(1/\varepsilon^2)+1} \in 2^{\mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)}$.

We define a mixed integer linear program $\mathrm{MILP}(\bar{T})$ in which configurations are chosen integrally and jobs are assigned fractionally to machine types. Note that we will call a solution of a MILP integral if both the integral and fractional variables have integral values. We introduce variables $z_{C,t} \in \mathbb{Z}_{\geq 0}$ for each machine type $t \in [K]$ and configuration $C \in \mathcal{C}_t(\bar{T})$, and $x_{j,t} \geq 0$ for each machine type $t \in [K]$ and job $j \in \mathcal{J}$.

7

For $\bar{p}_{tj} = \infty$ we set $x_{j,t} = 0$. Besides this, the MILP has the following constraints:

$$\sum_{C \in \mathcal{C}_t(\bar{T})} z_{C,t} = m_t \qquad\qquad \forall t \in [K] \quad (1)$$

$$\sum_{t \in [K]} x_{j,t} = 1 \qquad\qquad \forall j \in \mathcal{J} \quad (2)$$

$$\sum_{j \in J_t(p)} x_{j,t} \leq \sum_{C \in \mathcal{C}_t(\bar{T})} C_p z_{C,t} \qquad \forall t \in [K], p \in B_t \quad (3)$$

$$\sum_{C \in \mathcal{C}_t(\bar{T})} \mathrm{size}(C) z_{C,t} + \sum_{p \in S_t} p \sum_{j \in J_t(p)} x_{j,t} \leq m_t \bar{T} \qquad\qquad \forall t \in [K] \quad (4)$$

With constraint (1) the number of chosen configurations for each machine type equals the number of machines of this type. Due to constraint (2) the variables $x_{j,t}$ encode the fractional assignment of jobs to machine types. Moreover for each machine type it is ensured with constraint (3) that the summed up number of big jobs of each size is at most the number of big jobs that are used in the chosen configurations for the respective machine type. Lastly, (4) guarantees that the overall processing time of the configurations and small jobs assigned to a machine type does not exceed the area $m_t \bar{T}$. It is easy to see that the MILP models a relaxed version of the problem:

Lemma 6. *If there is schedule with makespan $\bar{T}$ there is a feasible (integral) solution of $\mathrm{MILP}(\bar{T})$, and if there is a feasible integral solution for $\mathrm{MILP}(\bar{T})$ there is a schedule with makespan at most $\bar{T} + \varepsilon^2 T$.*

Proof. Let σ be a schedule with makespan $\bar{T}$. Each machine of type t obeys exactly one configuration from $\mathcal{C}_t(\bar{T})$, and we set $z_{C,t}$ to be the number of machines of type t that obey C with respect to σ. Furthermore for a job j^* let t^* be the type of machine $\sigma(j^*)$. We set $x_{j^*,t^*} = 1$ and $x_{j^*,t} = 0$ for $t \neq t^*$. It is easy to check that all conditions are fulfilled.

Now let $(z_{C,t}, x_{j,t})$ be an integral solution of $\mathrm{MILP}(\bar{T})$. Using (2) we can assign the jobs to distinct machine types based on the $x_{j,t}$ variables. The $z_{C,t}$ variables can be used to assign configurations to machines such that each machine receives exactly one configuration using (1). Based on these configurations we can create slots for the big jobs and for each type t we can successively assign all of the big jobs assigned to this type to slots of the size of their processing time, because of (3). Now, for each type, we can iterate through the machines and greedily assign small jobs. When the makespan $\bar{T}$ is exceeded due to some job, we stop assigning to the current machine and continue with the next. Because of (4), all small jobs can be assigned in this fashion. Since the small jobs have size at most $\varepsilon^2 T$, we get a schedule with makespan at most $\bar{T} + \varepsilon^2 T$. $\qquad\square$
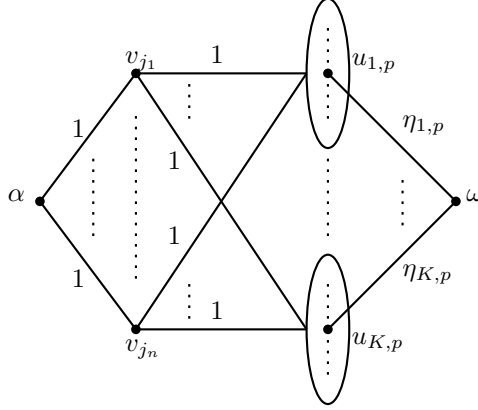
Figure 1: A sketch of the flow network.

We have $K2^{\mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)}$ integral variables, i.e., a constant number. Therefore MILP(T) can be solved in polynomial time, with the following classical result due to Lenstra [24] and Kannan [21]:

Theorem 7. *A mixed integer linear program with d integral variables and encoding size s can be solved in time $d^{\mathcal{O}(d)}\mathrm{poly}(s)$.*

Rounding. In this paragraph we describe how a feasible solution $(z_{C,t}, x_{j,t})$ for MILP($\bar{T}$) can be transformed into an integral feasible solution $(\bar{z}_{C,t}, \bar{x}_{j,t})$ for MILP($\bar{T}+\varepsilon T + \varepsilon^2 T$), where the second MILP is defined using the same configurations but accordingly changed right hand side. This is achieved via a flow network utilizing flow integrality.

For any (small or big) processing time p let $\eta_{t,p} = \lceil \sum_{j \in J_t(p)} x_{j,t} \rceil$ be the rounded up (fractional) number of jobs with processing time p that are assigned to machine type t. Note that for big job sizes $p \in B_t$, we have $\eta_{t,p} \leq \sum_{C \in \mathcal{C}_t(\bar{T})} C_p z_{C,t}$, because of (3) and because the right hand side is an integer.

Now we describe the flow network $G = (V, E)$ with source α and sink ω. For each job $j \in \mathcal{J}$ there is a job node v_j and an edge (α, v_j) with capacity 1 connecting the source and the job node. Moreover, for each machine type t we have processing time nodes $u_{t,p}$ for each processing time $p \in B_t \cup S_t$. The processing time nodes are connected to the sink via edges $(u_{t,p}, \omega)$ with capacity $\eta_{t,p}$. Lastly, for each job j and machine type t with $\bar{p}_{t,j} < \infty$, we have an edge $(v_j, u_{t,\bar{p}_{t,j}})$ with capacity 1 connecting the job node with the corresponding processing time nodes. We outline the construction in Figure 1. Obviously we have $|V| \leq (K+1)n+2$ and $|E| \leq (2K+1)n$.

Lemma 8. *G has a maximum flow with value n.*

Proof. Since the outgoing edges from α have summed up capacity n, n is a trivial upper bound for the maximum flow. The solution $(z_{C,t}, x_{j,t})$ for $\text{MILP}(\bar{T})$ can be used to design a flow f with value n, by setting $f((\alpha, v_j)) = 1$, $f((v_j, u_{t, \bar{p}_{t,j}})) = x_{j,t}$ and $f((u_{t,y}, \omega)) = \sum_{j \in J_t(y)} x_{j,t}$. It is easy to check that f is indeed a feasible flow with value n. $\qquad\square$

Using the Ford-Fulkerson algorithm, an integral maximum flow f^* can be found in time $\mathcal{O}(|E|f^*) = \mathcal{O}(Kn^2)$. Due to flow conservation, for each job j there is exactly one machine type t^* such that $f((v_j, u_{t^*, \bar{p}_{t^*,j}})) = 1$, and we set $\bar{x}_{j,t^*} = 1$ and $\bar{x}_{j,t} = 0$ for $t \neq t^*$. Moreover, we set $\bar{z}_{C,t} = z_{C,t}$. Obviously $(\bar{z}_{C,t}, \bar{x}_{j,t})$ fulfils (1) and (2). Furthermore, (3) is fulfilled, because of the capacities and because $\eta_{t,p} \leq \sum_{C \in \mathcal{C}_t(\bar{T})} C_p z_{C,t}$ for big job sizes p. Utilizing the geometric rounding and the convergence of the geometric series, as well as $\sum_{j \in J_t(p)} \bar{x}_{j,t} \leq \eta_{t,p} < \sum_{j \in J_t(p)} x_{j,t} + 1$, we get:

$$\sum_{p \in S_t} p \sum_{j \in J_t(p)} \bar{x}_{j,t} < \sum_{p \in S_t} p \sum_{j \in J_t(p)} x_{j,t} + \sum_{p \in S_t} p < \sum_{p \in S_t} p \sum_{j \in J_t(p)} x_{j,t} + \varepsilon^2 T \frac{1 + \varepsilon}{\varepsilon}$$

Hence, we have $\sum_{C \in \mathcal{C}_t(\bar{T})} \text{size}(C) \bar{z}_{C,t} + \sum_{p \in S_t} p \sum_{j \in J_{t,s}} \bar{x}_{j,t} < m_t(\bar{T} + \varepsilon T + \varepsilon^2 T)$ and therefore (4) is fulfilled as well.

Analysis. The solution found for $\text{MILP}(\bar{T})$ can be turned into an integral solution for $\text{MILP}(\bar{T} + \varepsilon T + \varepsilon^2 T)$. Like described in the proof of Lemma 6 this can easily be turned into a schedule with makespan $\bar{T} + \varepsilon T + \varepsilon^2 T + \varepsilon^2 T \leq (1 + 4\varepsilon)T$. It is easy to see that the running time of the algorithm by Lenstra and Kannan dominates the overall running time. Since $\text{MILP}(\bar{T})$ has $\mathcal{O}(K/\varepsilon \log 1/\varepsilon + n)$ many constraints, Kn fractional and $K2^{\mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)}$ integral variables, the running time of the algorithm can be bounded by:

$$(K2^{\mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)})^{\mathcal{O}(K2^{\mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)})} \text{poly}((K/\varepsilon \log 1/\varepsilon)|I|) = 2^{K2^{\mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)}} \text{poly}(|I|)$$

3 Better running time

We improve the running time of the algorithm using techniques that utilize results concerning the existence of solutions for integer linear programs (ILPs) with a certain simple structure. In a first step we can reduce the running time to be only singly exponential in $1/\varepsilon$ with a technique by Jansen [18]. Then we further improve the running time to the one claimed in Theorem 1 with a very recent result by Jansen,

Klein and Verschae [19]. Both techniques rely upon the following result about integer cones by Eisenbrandt and Shmonin [12].

Theorem 9. *Let $X \subset \mathbb{Z}^d$ be a finite set of integer vectors and let $b \in \text{int-cone}(X) = \{\sum_{x \in X} \lambda_x x \mid \lambda_x \in \mathbb{Z}_{\geq 0}\}$. Then there is a subset $\tilde{X} \subseteq X$, such that $b \in \text{int-cone}(\tilde{X})$ and $|\tilde{X}| \leq 2d \log(4dM)$, with $M = \max_{x \in X} \|x\|_\infty$.*

For the first improvement of the running time, this theorem is used to show:

Corollary 10. $\text{MILP}(\bar{T})$ *has a feasible solution, where for each machine type at most $\mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)$ of the corresponding integer variables are non-zero.*

We get the better running time by guessing the non-zero variables and removing all the others from the MILP. The number of possibilities of choosing $\mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)$ elements out of a set of $2^{\mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)}$ elements can be bounded by $2^{\mathcal{O}(1/\varepsilon^2 \log^4 1/\varepsilon)}$. Considering all the machine types we can bound the number of guesses by $2^{\mathcal{O}(K/\varepsilon^2 \log^4 1/\varepsilon)}$. The running time of the algorithm by Lenstra and Kannan with $\mathcal{O}(K/\varepsilon \log^2 1/\varepsilon)$ integer variables can be bounded by:

$$\mathcal{O}(K/\varepsilon \log^2 1/\varepsilon)^{\mathcal{O}(K/\varepsilon \log^2 1/\varepsilon)} \text{poly}(|I|) = 2^{\mathcal{O}(K \log(K) 1/\varepsilon \log^3 1/\varepsilon)} \text{poly}(|I|)$$

This yields a running time of:

$$2^{\mathcal{O}(K \log(K) 1/\varepsilon^2 \log^4 1/\varepsilon)} \text{poly}(|I|)$$

In the following we first proof Corollary 10 and then introduce the technique from [19] to further reduce the running time.

Proof of Corollary 10.   We consider the so called *configuration ILP* for scheduling on identical machines. Let m' be a given number of machines, P be a set of processing times with multiplicities $k_p \in \mathbb{Z}_{>0}$ for each $p \in P$ and let $\mathcal{C} \subseteq \mathbb{Z}_{\geq 0}^P$ be some finite set of configurations for P. The configuration ILP for m', P, $k = (k_p)_{p \in P}$, and $\mathcal{C}$ is given by:

$$\sum_{C \in \mathcal{C}} C_p y_C = k_p \qquad\qquad \forall p \in P \qquad\qquad (5)$$

$$\sum_{C \in \mathcal{C}} y_C = m' \qquad\qquad (6)$$

$$y_C \in \mathbb{Z}_{\geq 0} \qquad\qquad \forall C \in \mathcal{C} \qquad\qquad (7)$$

The default case that we will consider most of the time is that $\mathcal{C}$ is given by a target makespan T that upper bounds the size of the configurations.

11

Let's assume we had a feasible solution $(\tilde{z}_{C,t}, \tilde{x}_{j,t})$ for MILP$(\bar{T})$. For $t \in [K]$ and $p \in B_t$ we set $\tilde{k}_{t,p} = \sum_{C \in \mathcal{C}_t(\bar{T})} C_p \tilde{z}_{C,t}$. We fix a machine type t. By setting $y_C = \tilde{z}_{C,t}$, we get a feasible solution for the configuration ILP given by m_t, B_t, $\tilde{k}_t$ and $\mathcal{C}_t(\bar{T})$. Theorem 9 can be used to show the existence of a solution for the ILP with only a few non-zero variables: Let X be the set of column vectors corresponding to the left hand side of the ILP and b be the vector corresponding to the right hand side. Then $b \in \text{int-cone}(X)$ holds and Theorem 9 yields that there is a subset $\tilde{X}$ of X with cardinality at most $2(|B_t| + 1) \log(4(|B_t| + 1)1/\varepsilon^2) \in \mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)$ and $b \in \text{int-cone}(\tilde{X})$. Therefore there is a solution $(\breve{y}_C)$ for the ILP with $\mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)$ many non-zero variables. If we set $\breve{z}_{C,t} = \breve{y}_C$ and $\breve{x}_{j,t} = \tilde{x}_{j,t}$ and perform corresponding steps for each machine type, we get a solution $(\breve{z}_{C,t}, \breve{x}_{j,t})$ that obviously satisfies constraints (1),(2) and (3) of MILP$(\bar{T})$. The last constraint is also satisfied, because the number of covered big jobs of each size does not change and therefore the overall size of the configurations does not change either for each machine type. This completes the proof of Corollary 10.

Further Improvement of the Running Time. The main ingredient of the technique by Jansen et al. [19] is a result about the configuration ILP, for the case that there is a target makespan T' upper bounding the configuration sizes. Let $\mathcal{C}(T')$ be the set of configurations with size at most T'. We need some further notation. The *support* of any vector of numbers v is the set of indices with non-zero entries, i.e., $\text{supp}(v) = \{i \,|\, v_i \neq 0\}$. A configuration is called *simple*, if the size of its support is at most $\log(T' + 1)$, and *complex* otherwise. The set of complex configurations from $\mathcal{C}(T')$ is denoted by $\mathcal{C}^c(T')$.

Theorem 11. *Let the configuration ILP for m', P, k, and $\mathcal{C}(T')$ have a feasible solution and let both the makespan T' and the processing times from P be integral. Then there is a solution (y_C) for the ILP that satisfies the following conditions:*

(i) $|\text{supp}(y|_{\mathcal{C}^c(T')})| \leq 2(|P| + 1) \log(4(|P| + 1)T')$ *and* $y_C \leq 1$ *for* $C \in \mathcal{C}^c(T')$.
(ii) $|\text{supp}(y)| \leq 4(|P| + 1) \log(4(|P| + 1)T')$.

We will call such a solution *thin*. Furthermore they argue:

Remark 12. There are at most $2^{\mathcal{O}(\log^2(T') + \log^2(|P|))}$ simple configurations.

The better running time can be achieved by determining configurations that are equivalent to the complex configurations (via guessing and dynamic programming), guessing the support of the simple configurations, and solving the MILP with few integral variables. The approach is a direct adaptation of the one in [19] for our case. In the following, we explain the additional steps of the modified algorithm in more detail, analyse the running time and present an outline of the complete algorithm.

We have to ensure that the makespan and the processing times are integral and that the makespan is small. After the geometric rounding step we scale the makespan and the processing times, such that $T = 1/\varepsilon^3$ and $\bar{T} = (1+\varepsilon)/\varepsilon^3$ holds and the processing times have the form $(1+\varepsilon)^x \varepsilon^2 T = (1+\varepsilon)^x/\varepsilon$. Next we apply a second rounding step for the big processing times, setting $\breve{p}_{t,j} = \lceil \bar{p}_{t,j} \rceil$ for $\bar{p}_{t,j} \in B_t$ and denote the set of these processing times by $\breve{B}_t$. Obviously we have $|\breve{B}_t| \le |B_t| \le 1/\varepsilon \log(1/\varepsilon^2) + 1$. We denote the corresponding instance by $\breve{I}$. Since for a schedule with makespan T for instance I there are at most $1/\varepsilon^2$ big jobs on any machine, we get:

Lemma 13. *If there is a schedule with makespan at most T for I, the same schedule has makespan at most $(1+2\varepsilon)T$ for instance $\breve{I}$ and any schedule for instance $\breve{I}$ can be turned into a schedule for I without increase in the makespan.*

We set $\breve{T} = (1+2\varepsilon)T$ and for each machine type t we consider the set of configurations $\mathcal{C}_t(\lfloor \breve{T} \rfloor)$ for $\breve{B}_t$ with size at most $\lfloor \breve{T} \rfloor$. Rounding down $\breve{T}$ ensures integrality and causes no problems, because all big processing times are integral. Furthermore let $\mathcal{C}_t^c(\lfloor \breve{T} \rfloor)$ and $\mathcal{C}_t^s(\lfloor \breve{T} \rfloor)$ be the subsets of complex and simple configurations. Due to Remark 12 we have:

$$|\mathcal{C}_t^s(\lfloor \breve{T} \rfloor)| \in 2^{\mathcal{O}(\log^2 \lfloor \breve{T} \rfloor + \log^2 |\breve{B}_t|)} = 2^{\mathcal{O}(\log^2 1/\varepsilon))} \tag{8}$$

Due to Theorem 11 (using the same considerations concerning configuration ILPs like in the last paragraph), we get that there is a solution $(\breve{z}_C, \breve{x}_{j,t})$ for MILP$(\breve{T})$ (adjusted to this case) that uses for each machine type t at most $4(|\breve{B}_t|+1)\log(4(|\breve{B}_t|+1)\lfloor \breve{T} \rfloor) \in \mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)$ many configurations from $\mathcal{C}_t(\lfloor \breve{T} \rfloor)$. Moreover, at most $2(|\breve{B}_t| + 1)\log(4(|\breve{B}_t|+1)\lfloor \breve{T} \rfloor) \in \mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)$ complex configurations are used and each of them is used only once. Since each configuration corresponds to at most $1/\varepsilon^2$ jobs, there are at most $\mathcal{O}(1/\varepsilon^3 \log^2 1/\varepsilon)$ many jobs for each type corresponding to complex configurations. Hence, we can determine the number of complex configurations m_t^c for machine type t along with the number of jobs $k_{t,p}^c$ with processing time $p \in \breve{B}_t$ that are covered by a complex configuration in $(1/\varepsilon^3 \log^2 1/\varepsilon)^{\mathcal{O}(K/\varepsilon \log 1/\varepsilon)} = 2^{\mathcal{O}(K/\varepsilon \log^2 1/\varepsilon)}$ many steps via guessing. Now we can use a dynamic program to determine configurations (with multiplicities) that are equivalent to the complex configurations in the sense that their size is bounded by $\lfloor \breve{T} \rfloor$, their summed up number is m_t^c and they cover exactly $k_{t,p}^c$ jobs with processing time p. The dynamic program iterates through $[m_t^c]$ determining $\breve{B}_t$-indexed vectors y of non-negative integers with $y_p \le k_{t,p}^c$. A vector y computed at step i encodes that y_p jobs of size p can be covered by i configurations from $\mathcal{C}_t(\lfloor \breve{T} \rfloor)$. We denote the set of configurations the program computes with $\tilde{\mathcal{C}}_t$ and the multiplicities with $\tilde{z}_C$ for $C \in \tilde{\mathcal{C}}_t$. It is easy to see that the running time of such a program can be bounded by $\mathcal{O}(m_t^c (\prod_{p \in \breve{B}_t} (k_{t,p}^c + 1))^2)$. Using $m_t^c \in \mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)$ and

$k_{t,p}^c \in \mathcal{O}(1/\varepsilon^3 \log^2 1/\varepsilon)$ this yields a running time of $K2^{\mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)}$, when considering all the machine types.

Having determined configurations that are equivalent to the complex configurations, we may just guess the simple configurations. For each machine type, there are at most $2^{\mathcal{O}(\log^2 1/\varepsilon)}$ simple configurations and the number of configurations we need is bounded by $\mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)$. Therefore, the number of needed guesses is bounded by $2^{\mathcal{O}(K/\varepsilon \log^4 1/\varepsilon)}$. Now we can solve a modified version of MILP($\check{T}$) in which z_C is fixed to $\tilde{z}_C$ for $C \in \tilde{\mathcal{C}}_t$ and only variables $z_{C'}$ corresponding to the guessed simple configurations are used. The running time for the algorithm by Lenstra and Kannan can again be bounded by $2^{\mathcal{O}(K \log K 1/\varepsilon \log^3 1/\varepsilon)}\mathrm{poly}(|I|)$. Thus we get an overall running time of $2^{\mathcal{O}(K \log K 1/\varepsilon \log^4 1/\varepsilon)}\mathrm{poly}(|I|)$. Considering the two cases $2^{\mathcal{O}(K \log K 1/\varepsilon \log^4 1/\varepsilon)} < \mathrm{poly}(|I|)$ and $2^{\mathcal{O}(K \log K 1/\varepsilon \log^4 1/\varepsilon)} \geq \mathrm{poly}(|I|)$ yields the claimed running time of:

$$2^{\mathcal{O}(K \log(K)1/\varepsilon \log^4 1/\varepsilon)} + \mathrm{poly}(|I|)$$

Hence, the proof of the part of Theorem 1 concerning unrelated scheduling is complete. We conclude this section with a summary of the complete algorithm.

Algorithm 14.
   (i) Simplify the input via scaling, geometric rounding and a second rounding step for the big jobs with an error of $2\varepsilon T$. We now have $T = 1/\varepsilon^3$.
  (ii) Guess the number of machines m_t^c with a complex configuration for each machine type t along with the number $k_{t,p}^c$ of jobs with processing time p covered by complex configurations for each big processing time $p \in \check{B}_t$.
 (iii) For each machine type t determine via dynamic programming configurations that are equivalent to the complex configurations.
  (iv) Guess the simple configurations used in a thin solution.
   (v) Build the simplified mixed integer linear program MILP($\check{T}$) in which the variables for configurations from step (iii) are fixed and only integral variables for configurations guessed in step (iv) are used. Solve it with the algorithm by Lenstra and Kannan.
  (vi) If there is no solution for each of the guesses, report that there is no solution with makespan T.
 (vii) Generate an integral solution for MILP($\check{T} + \varepsilon T + \varepsilon^2 T$) via a flow network utilizing flow integrality.
(viii) With an additional error of $\varepsilon^2 T$ due to the small jobs the integral solution is turned into a schedule.

4 The Santa Claus Problem

Adapting the result for unrelated scheduling we achieve an EPTAS for the Santa Claus problem. It is based on the basic EPTAS together with the second running time improvement. In the following we show the needed adjustments.

Preliminaries. Wlog. we present a $(1 - \varepsilon)^{-1}$-approximation instead of a $(1 + \varepsilon)$-approximation. Moreover, we assume $\varepsilon < 1$ and that $m \leq n$, because otherwise the problem is trivial.

The dual approximation method can be applied in this case as well. However, since we have no approximation algorithm with a constant rate, the binary search is slightly more expensive. Still we can use for example the algorithm by Bezáková and Dani [4] to find a bound B for the optimal makespan with $B \leq \text{OPT} \leq (n - m + 1)B$. In $\mathcal{O}(\log((n - m)/\varepsilon))$ many steps we can find a guess for the optimal minimum machine load T^* such that $T^* \leq \text{OPT} < T^* + \varepsilon B$ and therefore $T^* > (1 - \varepsilon)\text{OPT}$. It suffices to find a procedure that given an instance and a guess T outputs a solution with objective value at least $(1 - \alpha\varepsilon)T$ for some constant α.

Concerning the simplification of the input, we first scale the makespan and the running times such that $T = 1/\varepsilon^3$. Then we set the processing times that are bigger than T equal to T. Next we round the processing times down via geometric rounding: We set $\bar{p}_{t,j} = (1 - \varepsilon)^x \varepsilon^2 T$ with $x = \lceil \log_{1-\varepsilon} p_{tj}/(\varepsilon^2 T) \rceil$. The number of big jobs for any machine type is again bounded by $1/\varepsilon \log(1/\varepsilon^2) \in \mathcal{O}(1/\varepsilon \log 1/\varepsilon)$. For the big jobs we apply the second rounding step setting $\breve{p}_{t,j} = \lfloor \bar{p}_{t,j} \rfloor$ and denote the resulting big processing times with $\breve{B}_t$, the corresponding instance by $\breve{I}$ and the occurring small processing times by S_t. The analogue of Lemma 13 holds, i.e. at the cost of $2\varepsilon T$ we may search for a solution for the rounded instance $\breve{I}$. We set $\breve{T} = (1 - 2\varepsilon)T$.

MILP. In the Santa Claus problem it makes sense to use configurations of size bigger than $\breve{T}$. Let $P = \lfloor \breve{T} \rfloor + \max\{\breve{p}_{t,j} \mid t \in [K], j \in \breve{B}_t\}$. It suffices to consider configurations with size at most P and for each machine type t we denote the corresponding set of configurations by $\mathcal{C}_t(P)$. Again we can bound $\mathcal{C}_t(P)$ by $2^{\mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)}$. The MILP has integral variables $z_{C,t}$ for each such configuration and fractional ones like before. The constraints (1) and (2) are adapted changing only the set of configurations and for constraint (3) additionally in this case the left-hand side has to be at least as big as the right hand side. The last constraint (4) has to be changed more. For this we partition $\mathcal{C}_t(P)$ into the set $\hat{\mathcal{C}}_t(P)$ of big configurations with size bigger than $\lfloor \breve{T} \rfloor$ and the set $\check{\mathcal{C}}_t(P)$ of small configurations with size at most $\lfloor \breve{T} \rfloor$. The

changed constraint has the following form:

$$\sum_{C \in \check{\mathcal{C}}_t(P)} \text{size}(C) z_{C,t} + \sum_{p \in S_t} p \sum_{j \in J_t(p)} x_{j,t} \geq (m_t - \sum_{C \in \hat{\mathcal{C}}_t(P)} z_{C,t}) \check{T} \qquad \forall t \in [K] \qquad (9)$$

We denote the resulting MILP by $\text{MILP}(\check{T}, P)$ and get the analogue of Lemma 6:

Lemma 15. *If there is schedule with minimum machine load $\check{T}$, there is a feasible (integral) solution of $\text{MILP}(\check{T}, P)$; and if there is a feasible integral solution for $\text{MILP}(\check{T}, P)$, there is a schedule with minimum machine load at least $\check{T} - \varepsilon^2 T$.*

Proof. Let σ be a schedule with minimum machine load $\check{T}$. We first consider only the machines for which the received load due to big jobs is at most P. These machines obey exactly one configuration from $\mathcal{C}_t(P)$ and we set the corresponding integral variables like before. The rest of the integral variables we initially set to 0. Now consider a machine of type t that receives more than P load due to big jobs. We can successively remove a biggest job from the set of big jobs assigned to the machine until we reach a subset with summed up processing time at most P and bigger than $\lfloor \check{T} \rfloor$. This set corresponds to a big configuration C' and we increment the variable $z_{C',t}$. The fractional variables are set like in the unrelated scheduling case and it is easy to verify that all constraints are satisfied.

Now let $(z_{C,t}, x_{j,t})$ be an integral solution of $\text{MILP}(\check{T})$. Again we can assign the jobs to distinct machine types based on the $x_{j,t}$ variables and the configurations to machines based on the $z_{C,t}$ variables such that each machine receives at most one configuration. Based on these configurations we can create slots for the big jobs and for each type t we can successively assign big jobs until all slots are filled. Now we can, for each type, iterate through the machines that received small configurations and greedily assign small jobs. When the makespan $\bar{T}$ would be exceeded due to some job, we stop assigning to the current machine (not adding the current job) and continue with the next machine. Because of (9) we can cover all of the machines by this. Since the small jobs have size at most $\varepsilon^2 T$ we get a schedule with makespan at least $\bar{T} - \varepsilon^2 T$. There may be some remaining jobs that can be assigned arbitrarily. $\square$

To solve the MILP we adapt the techniques by Jansen et al. [19], which is slightly more complicated for the modified MILP. Unlike in the previous section in order to get a thin solution that still fulfils (9), we have to consider big and small configurations separately for each machine type. Note that for a changed solution of the MILP (9) is fulfilled, if the summed-up size of the small and the summed up number of the big configurations is not changed. Given a solution $(\tilde{z}_{C,t}, \tilde{x}_{j,t})$ for the MILP and a machine type t, we set $\check{m}_t = \sum_{C \in \check{\mathcal{C}}_t(P)} \tilde{z}_{C,t}$ and $\hat{m}_t = \sum_{C \in \hat{\mathcal{C}}_t(P)} \tilde{z}_{C,t}$, and furthermore $\check{k}_{t,p} = \sum_{C \in \check{\mathcal{C}}_t(P)} C_p \tilde{z}_{C,t}$ and $\hat{k}_{t,p} = \sum_{C \in \hat{\mathcal{C}}_t(P)} C_p \tilde{z}_{C,t}$ for $p \in \check{B}_t$. We

get two configuration ILPs: The first is given by $\check{m}_t$, $\check{B}_t$, $\check{k}_t$ and $\check{\mathcal{C}}_t(P)$ and we call it the *small* ILP. The second is given by $\hat{m}_t$, $\check{B}_t$, $\hat{k}_t$ and $\hat{\mathcal{C}}_t(P)$ and we call it the *big* ILP. For the small ILP the set of configurations is given by the upper bound $\lfloor \check{T} \rfloor$ on the configuration size and we define the simple and complex configurations accordingly denoting them by $\check{\mathcal{C}}^s(P)$ and $\check{\mathcal{C}}^c(P)$ respectively. We can directly apply Theorem 11 to the small ILP like before without changing the summed-up size of the small configurations. This is not the case for the big ILP because in this case the set of configurations is defined by an upper and lower bound for the configuration size and hence Theorem 11 can not be applied directly. Note that considering the set of configurations given just by the upper bound P is not an option, since this could change the number of big configurations that are used. However, when looking more closely into the proof of Theorem 11 given in [19], it becomes apparent that the result can easily be adapted. For this we call a configuration C in this case simple if $|\mathrm{supp}(C)| \leq \log(P+1)$ and complex otherwise and denote the corresponding sets by $\hat{\mathcal{C}}^s(P)$ and $\hat{\mathcal{C}}^c(P)$ respectively. Without going into details we give the outline how the proof can be adjusted to this case:

The main tools in the proof are variations of Theorem 9 and the so called Sparsification Lemma. Theorem 9 actually works with any set of configurations and therefore we can restrict its use to big configuration. Moreover, the Sparsification Lemma is used to exchange complex configurations that are used multiple times with configurations that have a smaller support but the same size. Therefore big configurations are exchanged only with other big configurations. Moreover, the Sparsification Lemma still holds when considering a set of configurations with a lower and upper bound for the size.

Hence, there is a thin solution for the big ILP and obviously the summed-up number of configurations stays the same. Summarizing we get:

Corollary 16. *If MILP($\check{T}$) has a solution, there is also a solution $(z_{C,t}, x_{j,t})$ such that for each machine type t:*

(i) $|\mathrm{supp}(y|_{\check{\mathcal{C}}^c_t(P)})| \leq 2(|\check{B}_t| + 1) \log(4(|\check{B}_t| + 1)\lfloor \check{T} \rfloor)$, $|\mathrm{supp}(y|_{\hat{\mathcal{C}}^c_t(P)})| \leq 2(|\check{B}_t| + 1) \log(4(|\check{B}_t| + 1)P)$ *and* $z_{C,t} \leq 1$ *for* $C \in \check{\mathcal{C}}^c_t(P) \cup \hat{\mathcal{C}}^c_t(P)$.

(ii) $|\mathrm{supp}(z_t)| \leq 4(|\check{B}_t| + 1)(\log(4(|\check{B}_t| + 1)\lfloor \check{T} \rfloor) + \log(4(|\check{B}_t| + 1)P))$.

Note that like before the terms above can be bounded by $\mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)$. Utilizing this corollary we can again solve the MILP rather efficiently. For this we have to guess the numbers $\check{m}^c_t$ and $\hat{m}^c_t$ of machines that are covered by small and big complex configurations respectively. In addition we guess like before the numbers of big jobs corresponding to the complex configurations. With this we can determine via dynamic programming suitable configurations. For the small configurations we can use the same dynamic program as before and for the second one we can use a similar

17

one that guarantees that we find big configurations. In the MILP we fix the big configurations we have determined and guess the non-zero variables corresponding to the simple configurations. Although this procedure is a little bit more complicated than in the unrelated machine case, the bound for the running time remains the same.

Rounding. To get an integral solution of the MILP we build a similar flow network. However in this case $\eta_{t,p} = \lfloor \sum_{j \in J_t(p)} x_{j,t} \rfloor$ is set to be the rounded *down* (fractional) number of jobs with processing time p that are assigned to machine type t. We get $\eta_{t,p} \geq \sum_{C \in \mathcal{C}(T)} C_\ell z_{C,t}$ for big processing times p. The flow network looks basically the same, with one important difference: The $(u_{t,p}, \omega)$ have a *demand* of $\eta_{t,p}$ and an capacity of ∞. We may introduce demands of 0 for all the other edges. The analogue of Lemma 8 holds, that is, the flow network has a (feasible) maximum flow with value n. Given such a flow we can build a new solution for the MILP changing the $x_{j,t}$ variables based on the flow decreasing the load due to small jobs by at most $\varepsilon T + \varepsilon^2 T$.

Flow networks with demands can be solved with a two-phase approach that first finds a feasible flow and than augments the flow until a max flow is reached. The first problem can be reduced to a max flow problem without demands in a flow network that is rather similar to the original one with at most two additional nodes and $\mathcal{O}(|V|)$ additional edges. Flow integrality still can be used. For details we refer to [1]. The running time again can be bounded by $\mathcal{O}(Kn^2)$. Hence the overall running time of the algorithm is $2^{\mathcal{O}(K \log(K)1/\varepsilon \log^4 1/\varepsilon)} + \text{poly}(|I|)$, which concludes the proof of Theorem 1.

5 Uniform Machinetypes

We consider the problem of unrelated scheduling with a constant number K of uniform machine types. In this version of the problem the input is as follows: Each job has a size p_{tj} for each uniform machine type t and each machine i has a speed value s_i and a type t_i. The processing time of job j on machine i is given by $p_{ij} = p_{t_ij}/s_i$.

We present an EPTAS and it has the same basic structure as the ones presented so far. However, both the MILP and its rounding are considerably more complicated and can be seen as a combination of the techniques from Section 2 with ideas from [18]. Note that in this section we have taken less effort to get a small running time in order to keep the presentation of the result less technical.

We set $\mathcal{M}_t = \{i \in \mathcal{M} \mid t_i = t\}$ for each $t \in [K]$ and $s_{\max}^{(t)} = \max\{s_i \mid i \in \mathcal{M}_t\}$. In the following, we refer to uniform machine types as machine types or just types.

Preliminaries. Again, we may assume that a target makespan T for instance I is given and we employ geometric rounding to both the job sizes and machine speeds. More precisely, if a job j has a size bigger than $Ts_{\max}^{(t)}$ for a machine type $t \in [K]$, we set $\bar{p}_{tj} = \infty$. For each job j with $p_{tj} < \infty$, we set $\bar{p}_{tj} = (1+\varepsilon)^x \varepsilon^2 Ts_{\max}^{(t)}$ with $x = \lceil \log_{1+\varepsilon}(p_{tj}/(\varepsilon^2 Ts_{\max}^{(t)})) \rceil$. Moreover, we set $\bar{s}_i = s_{\max}^{(t)}/(1+\varepsilon)^y$ with $y = \lceil \log_{1+\varepsilon}(s_{\max}^{(t)}/s_i) \rceil$ and call the rounded instance $\bar{I}$.

Lemma 17. *If there is a schedule with makespan at most T for I, the same schedule has makespan at most $(1+\varepsilon)^2 T$ for instance $\bar{I}$ and any schedule for instance $\bar{I}$ can be turned into a schedule for I without increase in the makespan.* $\qquad\square$

Therefore, it suffices to search for a schedule for instance $\bar{I}$ with makespan $\bar{T} := (1+\varepsilon)^2 T$. For the sake of simplicity, we do not use the ($\bar{\cdot}$)-notation in the following, i.e., we assume that the instance is already rounded and the makespan properly increased.

We fix some notation: A job size p is called *huge* for a speed s, if $p > Ts$; *big*, if $p \leq Ts$ and $p > \varepsilon^2 Ts$; and *small* otherwise. We will not consider assigning jobs on machines for whose speeds they are huge. For each machine type t, we denote the set of occurring speeds $\{s_i \mid i \in \mathcal{M}_t\}$ by V_t; the set of machines of type t and speed s by $\mathcal{M}_{t,s}$; and set $m_{t,s} := |\mathcal{M}_{t,s}|$. For each machine type t and speed s, let $S_{t,s}$ and $B_{t,s}$ be the sets of occurring small and big processing times. Furthermore, let P_t be the set of all occurring job sizes for type t. Like before, we have $|B_{t,s}| \in \mathcal{O}(1/\varepsilon \log 1/\varepsilon)$. For any processing time p we denote the set of jobs j with $p_{tj} = p$ by $J_t(p)$.

Separation of Machines. We will consider configurations for each machine type t and speed value $s \in V_t$. However, the number of distinct speed values could be dependent in m and we can not effort to introduce integral variables in the MILP for each of them. Instead, we will introduce integral variables only for the fastest speeds of each type and round the fractional variables. For the rounding approach, we will need a constant number of machines that receive some load from the slow speeds, and furthermore the speeds of these machines have to be faster than the slow speeds by some constant factor. This leads to a separation of the machines into three groups $G_{t,i}$ for $i \in [3]$ for each machine type t. This is done in a way, such that for $j > i$ the machines in group $G_{t,i}$ are faster than the ones in group $G_{t,j}$. For $i \in [3]$ and opt $\in \{\min, \max\}$, we set $s_{i,\mathrm{opt}}^{(t)} := \mathrm{opt}\{s_i \mid i \in G_{t,i}\}$. The partition is defined by two parameters. The first parameter

$$\kappa := \max\{|B_{t,s}| + 1 \mid \forall t \in [K], s \in V_t\} \in \mathcal{O}(1/\varepsilon \log 1/\varepsilon)$$

controls the number of machines in the first group and the second $\gamma := 1/\varepsilon^2$ the speed-gap between the first and the third group. More precisely:

- $G_{t,1}$ contains the κ fastest machines of type t.

- $G_{t,2}$ contains all machines of type t that are not contained in $G_{t,1}$ and whose speed is bigger than $\gamma s_{1,\min}^{(t)}$.

- $G_{t,3}$ contains the rest of the machines of type t.

Note that $G_{t,2}$ and $G_{t,3}$ might be empty. We denote the occurring speeds in group $G_{t,i}$ by $V_{t,i}$ and call the speeds from $V_{t,1} \cup V_{t,2}$ *fast* and the rest *slow*. With these definitions we have $(V_{t,1} \cup V_{t,2}) \cap V_{t,3} = \emptyset$ and $|V_{t,1}|, |V_{t,2}| \in \mathcal{O}(1/\varepsilon \log(1/\varepsilon))$, i.e., the fast and slow speed values are distinct and we have only a constant number of fast speed values.

MILP. For each machine type t and speed $s \in V_t$ we consider the set $\mathcal{C}_t(sT)$ of configurations C for the big processing times $B_{t,s}$ and with $\mathrm{size}(C) \leq sT$. Note that $|\mathcal{C}_t(sT)| \in 2^{\mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)}$ and therefore $|\bigcup_{s \in V_{t,1} \cup V_{t,2}} \mathcal{C}_t(sT)| \in 2^{\mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)}$.

The MILP formulation in this scenario follows the same basic ideas, but is more complicated than before. We assign jobs fractionally to machines types. For the fast machine speeds we chose configurations integrally and for the slow ones fractionally. Furthermore, we fractionally assign job sizes to machine speeds for which they are small. Lastly, we count the number of jobs of each job size that are assigned to each machine type. If the job size is big on some fast machine of the machine type, we require an integral number of jobs. More precisely, we introduce the following variables:

- Configuration variables $z_C^{(t,s)}$ for each machine type $t \in [K]$, occurring speed $s \in V_t$ and configuration $C \in \mathcal{C}_t(sT)$. If s is fast, we require $z_C^{(t,s)} \in \mathbb{Z}_{\geq 0}$ and otherwise $z_C^{(t,s)} \geq 0$.

- Job assignment variables $x_{j,t} \geq 0$ for each machine type $t \in [K]$ and job $j \in \mathcal{J}$.

- Job size assignment variables $y_{p,s}^{(t)} \geq 0$ for each machine type $t \in [K]$, speed $s \in V_t$ and job size $p \in S_{t,s}$.

- Counting variables $u_p^{(t)}$ for each machine type $t \in [K]$ and job size $p \in P_t$. If there is a fast speed $s \in V_{t,1} \cup V_{t,2}$, such that $p \in B_{t,s}$ we require $u_p^{(t)} \in \mathbb{Z}_{\geq 0}$ and otherwise $u_p^{(t)} \geq 0$.

Now the MILP is given by the above variables and the following constraints:

$$\sum_{C \in \mathcal{C}_t(sT)} z_C^{(t,s)} = m_{t,s} \qquad \forall t \in [K], s \in V_t \qquad (10)$$

$$\sum_{t \in [K]} x_{j,t} = 1 \qquad \forall j \in \mathcal{J} \qquad (11)$$

$$\sum_{j \in J_t(p)} x_{j,t} \leq u_p^{(t)} \qquad \forall t \in [K], p \in P_t \qquad (12)$$

$$\sum_{s:p \in B_{t,s}} \sum_{C \in \mathcal{C}_t(sT)} C_p z_C^{(t,s)} + \sum_{s:p \in S_{t,s}} y_{p,s}^{(t)} \geq u_p^{(t)} \qquad \forall t \in [K], p \in P_t \qquad (13)$$

$$\sum_{C \in \mathcal{C}_t(sT)} \text{size}(C) z_C^{(t,s)} + \sum_{p \in S_{t,s}} p y_{p,s}^{(t)} \leq m_{t,s} s T \qquad \forall t \in [K], s \in V_t \qquad (14)$$

The constraints (10) and (11) are very similar to constraints for the other MILPs that we consider. For each machine type it is ensured with the constraints (12) and (13) that the summed up number of jobs of each size is covered by the the chosen configurations and the small job assignments. Furthermore, (14) guarantees that the overall processing time of the configurations and small jobs assigned to a machine speed for each type does not exceed the available area.

Lemma 18. *If there is schedule with makespan T, there is a feasible (integral) solution of* $\text{MILP}(T)$*; and if there is a feasible integral solution for* $\text{MILP}(T)$*, there is a schedule with makespan at most* $(1 + \varepsilon^2)T$*.*

Proof. Given a schedule σ with makespan T, each machine of type t with speed s obeys exactly one configuration from $\mathcal{C}_t(sT)$ and we can set the variables $z_{C,t}$ accordingly. Furthermore, we set $x_{j,t_{\sigma(j)}} = 1$, $x_{j^*,t} = 0$ for $t \neq t_{\sigma(j)}$, $u_p^{(t)} := |\{j \,|\, t_{\sigma(j)} = t, p_{t,j} = p\}|$, and $y_{p,s}^{(t)} := |\{j \,|\, t_{\sigma(j)} = t, s_{\sigma(j)} = s, p_{t,j} = p\}|$. It is easy to check that all conditions are fulfilled.

Like we did in the proof of Lemma 6, given an integral solution (z, x, y, u) we can assign the jobs to machine types, and configurations to machines. Moreover, based one the $y_{p,s}^{(t)}$ variables we can assign jobs of size p that are assigned to type t to machines of speed s on which they are small. Because of (12) and (13) this can be done such that the remaining jobs of size p can be scheduled into slots provided by configurations. At this point each unscheduled job is assigned to a type and a speed. Utilizing (14), these jobs can be scheduled greedyly with an additive error of $\varepsilon^2 T$. $\qquad\square$

Lemma 19. *The MILP has $\mathcal{O}(K(n+m))$ many constraints, $\mathcal{O}(Knm)+m2^{\mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)}$ many variables and $K2^{\mathcal{O}(1/\varepsilon \log^2 1/\varepsilon)}$ integral variables. It can be solved in time:*

$$2^{\mathcal{O}(K\log(K)1/\varepsilon^3 \log^5 1/\varepsilon)}\mathrm{poly}(|I|)$$

Proof. The bounds for the number of constraints and variables are easy to verify using the above considerations as well as $|V_t| \leq m$ and $|P_t| \leq n$. The running time can be achieved with the first approach presented in Section 3: Using Theorem 9, we can argue that $\mathcal{O}(1/\varepsilon \log^2(1/\varepsilon))$ many integral variables for each machine type t and speed s suffice. Therefore the number of needed guesses is $2^{\mathcal{O}(K/\varepsilon^3 \log^5 1/\varepsilon)}$. Running the algorithm of Lenstra and Kannan with $\mathcal{O}(K/\varepsilon^2 \log^3(1/\varepsilon))$ many integral variables takes $2^{\mathcal{O}(K\log(K)1/\varepsilon^2 \log^4 1/\varepsilon)}\mathrm{poly}(|I|)$ time. Together we get the stated running time. $\qquad\square$

Rounding. We present rounding approaches for all fractional variables and start with the configuration variables.

Configuration Variables. We fix a type t and slow speed $s \in V_t$ and set $k_p := \sum_{C \in \mathcal{C}_t(sT)} C_p z_C^{(t,s)}$ for each $p \in B_{t,s}$. We have:

$$\sum_{C \in \mathcal{C}_t(sT)} z_C^{(t,s)} = m_{t,s} \tag{15}$$

$$\sum_{C \in \mathcal{C}_t(sT)} C_p z_C^{(t,s)} = k_p \qquad\qquad \forall p \in B_{t,s} \tag{16}$$

It is easy to check that, if we replace the solution of this LP with any other solution and change the MILP solution accordingly, the resulting MILP solution will still be feasible. We transform the solution into a basic feasible solution. This can be done in time polynomial in $1/\varepsilon$ and $|I|$. The LP has $|B_{t,s}| + 1$ many constraints and therefore the solution has at most $|B_{t,s}| + 1$ many variables greater than 0. Now the idea, is to round down the fractional values and to assign the respective job sizes that lost covering by the configurations to the fastest group $G_{t,1}$. More precisely, we chose some injective mapping ξ between the configurations C with fractional variables $z_C^{(t,s)}$ and the machines from $G_{t,1}$. This can be done, due to the choice of the parameter κ that regulates the number of machines in $G_{t,1}$. Now, we round down $z_C^{(t,s)}$ to the next integral value and increase $y_{p,s_{\xi(C)}}^{(t)}$ by $(z_C^{(t,s)} - \lfloor z_C^{(t,s)} \rfloor)C_p \leq C_p$ for each $p \in B_{t,s}$. We perform these steps for all types t and slow speeds $s \in V_t$. Note that any particular variable $y_{p,s}^{(t)}$ might be increased several times for each speed value for which p is big. Let $\tilde{y}_{p,s}^{(t)}$ denote the resulting increased $y_{p,s}^{(t)}$ variables and $\bar{z}_C^{(t,s)}$ the

22

resulting configuration variables. For $(\bar{z}, x, \tilde{y}, u)$ the constraints (10)-(12) obviously still hold and it is easy to see that this is also the case for (13), while (14) might be violated for speeds associated with the fastest machine groups. We show that a modified version of (14) still holds.

Consider a machine $i \in G_{t,1}$. For each slow speed value $s \in V_{t,3}$ there may be one configuration C that is mapped to i. The summed up job sizes that are reassigned to s_i because of this are bounded by sT. Summing up over all speed values $s \in V_{t,3}$ and utilizing the convergence of the geometric series, the rounding of the speed values, and the fact that $s_{3,\max}^{(t)} \leq \gamma s_{1,\max}^{(t)} = 1/\varepsilon^2 s_{1,\max}^{(t)}$, we get:

$$\sum_{s \in V_{t,3}} sT = T s_{3,\max}^{(t)} \sum_{s \in V_{t,3}} \frac{s}{s_{3,\max}^{(t)}} < T s_{3,\max}^{(t)} \sum_{i=0}^{\infty} \frac{1}{(1+\varepsilon)^i}$$

$$\leq T s_{1,\min}^{(t)}(\varepsilon + \varepsilon^2) \leq T s_i (\varepsilon + \varepsilon^2)$$

Hence, (14) holds if we increase the makespan on the right hand side by $(\varepsilon + \varepsilon^2)T$.

Counting Variables. The rounding step for the counting variables u is the easiest: We round them up and assign the extra job sizes to the fastest machine speed in the group, that is, for each $t \in [K]$ and $p \in P_t$ we set $\bar{u}_p^{(t)} = \lceil u_p^{(t)} \rceil$ and increase $\tilde{y}_{p,s^*}^{(t)}$ by $\bar{u}_p^{(t)} - u_p^{(t)} \leq 1$, where $s^* = s_{\max}^{(t)}$. We denote the changed $\tilde{y}_{p,s^*}^{(t)}$ by $\check{y}_{p,s^*}^{(t)}$ Again, it is easy to see that for $(\bar{z}, x, \check{y}, \bar{u})$ the constraints (10)-(13) still hold, while (14) is violated. However, we can bound the increase the fastest speed receives, again utilizing the geometric series:

$$\sum_{p \in S_{t,s^*}} p(\check{y}_{p,s^*}^{(t)} - \tilde{y}_{p,s^*}^{(t)}) \leq \sum_{p \in S_{t,s^*}} p \leq \varepsilon^2 s^* T \frac{1+\varepsilon}{\varepsilon} = (\varepsilon + \varepsilon^2) s^* T$$

Hence, (14) holds if we further increase the makespan by $(\varepsilon + \varepsilon^2)T$.

Job Size Assignment Variables. Consider the constraint (13) for the solution $(\bar{z}, x, \check{y}, \bar{u})$. Note that the right hand side and the first sum on the left hand side are both integral. Therefore, we can scale the $\check{y}_{p,s}^{(t)}$ variables down, such that $\sum_{s:p \in S_{t,s}} \check{y}_{p,s}^{(t)}$ is integral for each machine type t and job size $p \in P_t$ and (13) is still fulfilled. We fix a machine type t, set $k_p := \sum_{s:p \in S_{t,s}} \check{y}_{p,s}^{(t)}$ for each $p \in P_t$ and assume $k_p \in \mathbb{Z}$, because of the argument above. Furthermore, we set $L_{t,s} := \sum_{p \in S_{t,s}} p \check{y}_{p,s}^{(t)}$ for each $s \in V_t$.

With these definitions, we have:

$$\sum_{s:p\in S_{t,s}} \breve{y}_{p,s}^{(t)} = k_p \qquad\qquad \forall p \in P_t \qquad (17)$$

$$\sum_{p\in S_{t,s}} p\breve{y}_{p,s}^{(t)} = L_{t,s} \qquad\qquad \forall s \in V_t \qquad (18)$$

If we replace the values $\breve{y}_{p,s}^{(t)}$ with any other solution for the above LP, we get an equivalent MILP solution. We can use a variation of the classical rounding approach by Lenstra, Shmoys and Tardos [23] to transform the solution $\breve{y}_{p,s}^{(t)}$.

For the sake of completeness, we summarize the main ideas of the rounding. The solution is transformed into a basic feasible one and the following bipartite graph is considered. There are two types of nodes, some associated with sizes p and some with speeds s. For any p or s, there can be at most one node; there are such nodes if and only if there are fractional variable $\breve{y}_{p,s'}^{(t)}$ or $\breve{y}_{p',s}^{(t)}$ left; and they are connected with an edge, if there is a fractional variable $\breve{y}_{p,s}^{(t)}$. Using a counting argument and some further considerations, it can be shown that this graph is a pseudoforest, i.e., all connected components are either trees or trees with one extra edge. Furthermore, because k_p is integral, the definition of the graph, together with the constraint (17), yield that all the leafs are associated to speeds. Using this structure, we can define an injective mapping ξ from the job sizes for which there is a fractional variable $\breve{y}_{p,s}^{(t)}$ to the speeds such that $\breve{y}_{p,\xi(s)}^{(t)}$ is one of the fractional variables. This can be done as follows: For each connected component there may be at most one cycle in the graph with alternating size and speed nodes and a suitable injective mapping for the corresponding sizes and speeds can easily be found, by going around the cycle and appropriately mapping consecutive nodes. After removing the corresponding nodes and edges, only trees remain in the graph. For each tree we can chose an arbitrary leaf. The leaf corresponds to a speed and its neighbor to a size and we can map the size to the speed and remove both corresponding nodes from the graph. Iterating this yields the mapping ξ. All the above steps can be performed in polynomial time in $1/\varepsilon$ and $|I|$.

We use the mapping ξ to round the variables $\breve{y}_{p,s}^{(t)}$ and because ξ is injective we can guarantee that each speed receives at most one extra small job. More precisely, for each $s \in V_t$ we set $\bar{y}_{p,s}^{(t)} = \lceil \breve{y}_{p,s}^{(t)} \rceil$, if $\xi(p) = s$ and $\bar{y}_{p,s}^{(t)} = \lfloor \breve{y}_{p,s}^{(t)} \rfloor$ otherwise. For the solution $(\bar{z}, x, \bar{y}, \bar{u})$ the constraints (10)-(13) still hold, while for (14) the makespan has to be increased further by $\varepsilon^2 T$.

Job Assignment Variables. The rounding of the job assignment variables is the same as in the regular machine types case. The only difference is that we can set

24

$\eta_{t,p} = \bar{u}_p^{(t)}$ in this case. Since all the values $\bar{u}_p^{(t)}$ are integral in this case there is no rounding error in this step. Let $\bar{x}_{j,t}$ be the rounded version of $x_{j,t}$.

Summarizing the rounding steps, for the solution $(\bar{z}, \bar{x}, \bar{y}, \bar{u})$ the constraints (10)-(13) hold together with:

$$\sum_{C \in \mathcal{C}_t(sT)} \text{size}(C)\bar{z}_C^{(t,s)} + \sum_{p \in S_{t,s}} p\bar{y}_{p,s}^{(t)} \leq m_{t,s}s(1 + 2\varepsilon + 3\varepsilon^2)T \quad \forall t \in [K], s \in V_t \quad (19)$$

Analysis. Summarizing the above steps, we can construct a schedule with makespan at most $(1 + \varepsilon)^2(1 + 2\varepsilon + 4\varepsilon^2)T \leq (1 + 27\varepsilon)T$ (assuming a schedule with makespan T exists), by building and solving the MILP, then rounding it and lastly transforming it into a schedule like in the proof of Lemma 18. Solving the MILP is again the most expensive step and with a simple case analysis we get a running time of:

$$2^{\mathcal{O}(K \log(K)^{1/\varepsilon^3} \log^5 {1/\varepsilon})} + \text{poly}(|I|)$$

6   Vector Scheduling

We present an EPTAS for D-dimensional unrelated vector scheduling, where both the dimension D and the number K of machine types are constant. In this problem variant for each job j a D-dimensional processing time vector $p_{tj} = (p_{tj}^{(1)}, \ldots, p_{tj}^{(D)})$ is given and the makespan is defined as the maximum load any machine receives in any dimension, i.e., $C_{\max}(\sigma) = \max_{i \in \mathcal{M}} \|\sum_{j \in \sigma^{-1}(i)} p_{ij}\|_\infty$. We define $P = \{p_{tj}^{(d)} \mid d \in [D], t \in [K], j \in \mathcal{J}\}$ and $\mathcal{P} = \{p_{tj} \mid t \in [K], j \in \mathcal{J}\}$.

The EPTAS is a direct adaptation of the one for the one dimensional case. In the following we briefly describe the needed extra steps and modification. Note, that we consider this result to be a proof of concept and took little effort to optimize the running time.

Preliminaries. We again use the dual approximation approach to get a guess T of the makespan. As an upper bound for this we can use the schedule that we get by assigning each job j to a machine i where $\sum_{d=1}^{D} p_{ij}^{(d)}$ is minimal. It is easy to see that this approach yields a Dm-approximation and we can use this result for the dual approximation like described in Section 4.

First, we perform rounding steps similar to those for the other results. For each $p_{tj} \in \mathcal{P}$ with $p_{tj}^{(d)} > T$ in at least one dimension d we set $\bar{p}_{tj} = (\infty, \ldots, \infty)$ and for all other processing time vectors p_{tj} we apply geometric rounding. Let $\theta = (\varepsilon^2/D)^D$ be some threshold parameter. We set $\bar{p}_{tj}^{(d)} = (1 + \varepsilon)^x \theta T$ with $x = \lceil \log_{1+\varepsilon}(p_{tj}^{(d)}/(\theta T)) \rceil$ yielding a rounded vector $\bar{p}_{tj}$ and a corresponding rounded instance $\bar{I}$.

For a given processing time vector the numbers that can occur in the different dimensions may still differ strongly. This complicates the problem, but we can reduce the extra complexity to some degree via a second rounding step: For each $\bar{p}_{tj}$ we set $\tilde{p}_{tj}^{(d)} = \max\{\bar{p}_{tj}^{(d)}, \|\bar{p}_{tj}\|_\infty \varepsilon/D\}$ yielding a rounded vector $\tilde{p}_{tj}$ and a corresponding rounded instance $\tilde{I}$. Similar rounding steps were used by Chekuri and Khanna [8] and Bonifaci and Wiese [6].

Lemma 20. *If there is a schedule with makespan at most T for I, the same schedule has makespan at most $(1 + \varepsilon)^2 T$ for instance $\tilde{I}$ and any schedule for instance $\tilde{I}$ can be turned into a schedule for I without increase in the makespan.*

Proof. Consider a schedule σ with makespan T for I. The first rounding step may increase the makespan by a factor of $(1 + \varepsilon)$. We fix a machine i, and a dimension d and bound the increase in load on machine i in dimension d for instance $\tilde{I}$. Let j be a job with $\sigma(j) = i$. If $\bar{p}_{ij}^{(d)} = \|\bar{p}_{ij}\|_\infty$, then job j causes no extra load on i in dimension d and if $\bar{p}_{ij}^{(d')} = \|\bar{p}_{ij}\|_\infty$ for some dimension $d' \neq d$, there might be an increase of at most $\|\bar{p}_{ij}\|_\infty \varepsilon/D$. In fact, the summed up load machine i receives in dimension d' might increase the load in d by an ε/D-factor in this fashion. Because the load in dimension d' is bounded by $(1+\varepsilon)T$ and there are $D-1$ dimensions $d' \neq d$, the overall load increase in dimension d on i can be up to $(D - 1)(1 + \varepsilon)T\varepsilon/D \leq \varepsilon(1 + \varepsilon)T$. $\square$

Hence, we may search for a schedule for instance $\tilde{I}$ with makespan $\tilde{T} := (1+\varepsilon)^2 T$. For the sake of simplicity, we do not use the $(\tilde{\cdot})$-notation in the following, i.e., we assume that the instance is already rounded and the makespan properly increased.

In this context we call a size $q \in P$ *big*, if $q > \theta T$ and *small* otherwise. Furthermore, we call a processing time vector $p \in \mathcal{P}$ *big*, if there is a dimension $d \in [D]$, such that $p^{(d)}$ is big, and *small* otherwise. Because of the second rounding step, we have $p^{(d)} > \theta T \varepsilon/D$ for each big vector p and dimension d. Let B_t and S_t be the sets of big and small processing time vectors occurring on machine type t. Note that $|B_t| \leq (\lceil \log_{1+\varepsilon}(D/(\theta\varepsilon)) \rceil)^D \leq (3D/\varepsilon \log(D/\varepsilon))^D$. Using these definition, the bound on the number of big jobs is much bigger than in the other cases. We chose this definition, because in the rounding of the MILP solution, each machine may receive a big (but constant) number of jobs for each small job size and to bound the overall load the small jobs have to be appropriately small.

For each processing time vector $p \in \mathcal{P}$ we denote the set of jobs j with $p_{tj} = p$ with $J_t(p)$.

MILP. Similar to the one dimensional case, for any set V of processing time vectors we call the V-indexed vectors of non-negative integers $\mathbb{Z}_{\geq 0}^V$ configurations (for V), set the size size(C) of a configuration C to be the corresponding vector of sizes, i.e.,

$\text{size}(C) = \sum_{p \in \mathcal{P}} C_p p$, and set $\mathcal{C}_t(T)$ to be the set of configurations C for B_t with $\text{size}(C) \leq T^D$. Note that:

$$|\mathcal{C}_t(T)| \leq (\frac{D}{\theta\varepsilon} + 1)^{(3D/\varepsilon \log(D/\varepsilon))^D} \leq (\frac{D}{\varepsilon})^{3D(3D/\varepsilon \log D/\varepsilon)^D} \leq 2^{(3D/\varepsilon \log D/\varepsilon)^{D+1}}$$

The MILP is a straight-forward adaptation of the one for the one-dimensional case with one important difference: The jobs are fractionally assigned to configurations belonging to a type, instead of just being assigned to machine types. More precisely, we introduce integral variables $z_{C,t} \in \mathbb{Z}_{\geq 0}$ for each machine type $t \in [K]$ and configuration $C \in \mathcal{C}_t(T)$, and fractional variables $x_{j,t,C} \geq 0$ for each job $j \in \mathcal{J}$, machine type $t \in [K]$ and configuration $C \in \mathcal{C}_t(T)$. For $p_{tj} = \infty^D$ we set $x_{j,t,C} = 0$. $\text{MILP}(T)$ is given by:

$$\sum_{C \in \mathcal{C}_t(T)} z_{C,t} = m_t \qquad \forall t \in [K] \qquad (20)$$

$$\sum_{t \in [K]} \sum_{C \in \mathcal{C}_t(T)} x_{j,t,C} = 1 \qquad \forall j \in \mathcal{J} \qquad (21)$$

$$\sum_{j \in J_t(p)} x_{j,t,C} \leq C_p z_{C,t} \qquad \forall t \in [K], p \in B_t, C \in \mathcal{C}_t(T) \qquad (22)$$

$$\sum_{p \in S_t} p \sum_{j \in J_t(p)} x_{j,t,C} \leq (T^D - \text{size}(C)) z_{C,t} \qquad \forall t \in [K], C \in \mathcal{C}_t(T) \qquad (23)$$

Note that the last constraint is D-dimensional. Unlike in the other cases we can not transform an integral solution for $\text{MILP}(T)$ directly into a schedule with only a small increase in the makespan. However, we deal with this in the rounding step and still have:

Lemma 21. *If there is schedule with makespan T there is a feasible (integral) solution of* $\text{MILP}(T)$. $\qquad\square$

Using the algorithm by Lenstra and Kannan we can solve $\text{MILP}(T)$ in time $f(1/\varepsilon, D, K)\text{poly}(|I|)$ for some computable function f.

Rounding. Using a variation of the rounding approach for the one dimensional case we can transform a solution (z, x) for $\text{MILP}(T)$ into a schedule with a makespan of at most $(1 + \varepsilon + \varepsilon^2)T$. The main difference is that we create nodes for pairs of *machines* and processing time vectors instead of pairs of *machine types* and processing times.

For each type t we assign configurations to machines of type t such that for each configuration $C \in \mathcal{C}_t(T)$ exactly $z_{C,t}$ configurations get assigned. Therefore, we can

assume that for each machine i a configuration $C^{(i)}$ is given. Based on this we can fractionally assign jobs to machines by setting $x_{j,i} = x_{j,t,C^{(i)}}/z_{C^{(i)},t}$, yielding:

$$\sum_{j \in \mathcal{J}} p_{ij} x_{j,i} \leq T^D \tag{24}$$

For each machine let $\mathcal{P}_i$ be the set of occurring processing time vectors for machine i, that is, for each $p \in \mathcal{P}$, we have $p \in \mathcal{P}_i$, iff there is a job j with $p_{ij} = p$ and $x_{j,i} > 0$. We set $\eta_{i,p} = \lceil \sum_{j \in J_t(p)} x_{j,i} \rceil$. If p is big, we have $\eta_{i,p} \leq C_p^{(i)}$, because of constraint (22).

Like in the one dimensional case, the flow network $G = (V, E)$ has a source α and sink ω, and for each job $j \in \mathcal{J}$ there is a job node v_j and an edge (α, v_j) with capacity 1 connecting the source and the job node. Moreover, for each machine i we have processing time vector nodes $u_{i,p}$ for each $p \in \mathcal{P}_i$. The processing time nodes are connected to the sink via edges $(u_{i,p}, \omega)$ with capacity $\eta_{i,p}$. Lastly, for each job j and machine type i with $x_{j,i} > 0$, we have an edge $(v_j, u_{i,p_{i,j}})$ with capacity 1 connecting the job node with the corresponding processing time vector nodes. The variables $x_{j,i}$ yield a flow with value n that is guaranteed to be correct because of the constraints of the MILP.

Lemma 22. *G has a maximum flow with value n.* □

Using the Ford-Fulkerson algorithm, an integral maximum flow f^* can be found in time $\mathcal{O}(|E|f^*) = \mathcal{O}(n^2 m)$. Due to flow conservation, for each job j there is exactly one machine i^* such that $f((v_j, u_{i^*,p_{i^*j}})) = 1$, and we set $\sigma(j) = i^*$. Analogously to the one dimensional case, for each big processing time vector p the schedule σ assigns at most $C_p^{(i)}$ many jobs j with $p_{ij} = p$ to machine i and for each small processing time p' vector one additional job j with $p_{ij} = p'$ may be assigned to i. Because of the choice of the parameter θ and the second rounding step, we can bound the extra load i receives.

Lemma 23. *Let $q \in P$ be small and $d \in [D]$. There are at most $(2\lceil \log_{1+\varepsilon}(D/\varepsilon) \rceil)^{D-1}$ processing time vectors $p \in \mathcal{P}$ with $p^{(d)} = q$.*

Proof. Let p be such a vector, $d' \neq d$ and $q' = p^{(d')}$. Because of the second rounding step we have $q' \geq q\varepsilon/D$ and $q \geq q'\varepsilon/D$ $p^{(d')}$. Now, because of the first rounding step there are only few such processing times q, more precisely at most $2\lceil \log_{1+\varepsilon}(D/\varepsilon) \rceil$. Hence, there can be at most $(2\lceil \log_{1+\varepsilon}(D/\varepsilon) \rceil)^{D-1}$ many such processing time vectors p. □

Using this lemma and the same argumentation as in the one dimensional case,

we can bound the extra load machine i receives in dimension d by:

$$(2\lceil \log_{1+\varepsilon}(D/\varepsilon) \rceil)^{D-1} \times \theta T \times \sum_{i=0}^{\infty} 1/(1+\varepsilon)^i$$
$$\leq 2(1/\varepsilon \log((D-1)/\varepsilon) + 1)^{D-1} \times (\varepsilon^2/D)^D T \times (1+\varepsilon)/\varepsilon$$
$$\leq 2(D/\varepsilon^2)^{D-1} \times (\varepsilon^2/D)^D T \times (1+\varepsilon)/\varepsilon$$
$$\leq \varepsilon^2 T \times (1+\varepsilon)/\varepsilon \leq (\varepsilon+\varepsilon^2)T$$

Summarizing, we have:

Lemma 24. *A solution* (z,x) *for* $\mathrm{MILP}(T)$ *can be transformed into a schedule with makespan at most* $(1+\varepsilon+\varepsilon^2)T$ *in time polynomial in* $|I|$ *and* $1/\varepsilon$.

Therefore there is an EPTAS for this case as well.

7 Conclusion

We presented efficient approximation schemes for several variants of the problem of scheduling on unrelated parallel machines. In the following, we briefly discuss some possible directions for further studies.

Better Running Times.   The presented approximation schemes have running times of the form $f(1/\varepsilon, K) + \mathrm{poly}(|I|)$ (or $f(1/\varepsilon, K, D) + \mathrm{poly}(|I|)$ in the vector scheduling case). While we took some effort to optimize f at least for the first two schemes, we did not optimize the $\mathrm{poly}(|I|)$ part in any of the results. Furthermore, for the case with a constant number of uniform types, one could study whether a quadratic or linear dependence in $1/\varepsilon$ (ignoring polylogarithmic dependencies) in the exponent of the $f(1/\varepsilon, K)$ part can be achieved, e.g. by utilizing techniques from [18] and [19]. Lastly, the EPTAS for the vector scheduling variant is basically just a proof of concept and we did not optimize the running time at all.

Lower Bound.   Chen, Ye and Zhang [11] showed that we can not hope for an EPTAS with a sub-linear dependency in $1/\varepsilon$ in the exponent, unless the exponential time hypothesis fails. It is unclear what can be ruled out in terms of the parameter K.

Job Types.   In the introduction we mentioned the concept of job types for scheduling on unrelated parallel machines: Two jobs j, j' are of the same type, if they behave the same on every machine i, i.e., $p_{ij} = p_{ij'}$. It is unknown, whether there is a PTAS

for scheduling on unrelated parallel machines with a constant number of job types. Furthermore, it is unknown, whether this problem is NP-hard. Indeed, the problem is in P for important special cases: For scheduling on identical parallel machines the number of job types is equal to the number of distinct processing times and for the case of the restricted assignment problem—where each job j has a size p_j and its processing time p_{ij} on machine i is either p_j or ∞—the number of distinct processing times is bounded by the number of job types and a constant number of job types implies a constant number of machine types. Both problems can be solved in polynomial time, if the number of distinct processing times is constant.

References

[1] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. Network flows: theory, algorithms, and applications. 1993.

[2] Arash Asadpour and Amin Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. *SIAM Journal on Computing*, 39(7):2970–2989, 2010.

[3] MohammadHossein Bateni, Moses Charikar, and Venkatesan Guruswami. Maxmin allocation via degree lower-bounded arborescences. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 543–552. ACM, 2009.

[4] Ivona Bezáková and Varsha Dani. Allocating indivisible goods. *ACM SIGecom Exchanges*, 5(3):11–18, 2005.

[5] Raphael Bleuse, Safia Kedad-Sidhoum, Florence Monna, Grégory Mounié, and Denis Trystram. Scheduling independent tasks on multi-cores with gpu accelerators. *Concurrency and Computation: Practice and Experience*, 27(6):1625–1638, 2015.

[6] Vincenzo Bonifaci and Andreas Wiese. Scheduling unrelated machines of few different types. *arXiv preprint arXiv:1205.0974*, 2012.

[7] Deeparnab Chakrabarty, Julia Chuzhoy, and Sanjeev Khanna. On allocating goods to maximize fairness. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 107–116. IEEE, 2009.

[8] Chandra Chekuri and Sanjeev Khanna. On multidimensional packing problems. *SIAM journal on computing*, 33(4):837–851, 2004.

[9] Lin Chen, Klaus Jansen, and Guochuan Zhang. On the optimality of approximation schemes for the classical scheduling problem. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 657–668. SIAM, 2014.

[10] Lin Chen, Dániel Marx, Deshi Ye, and Guochuan Zhang. Parameterized and approximation results for scheduling with a low rank processing time matrix. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 66. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[11] Lin Chen, Deshi Ye, and Guochuan Zhang. An improved lower bound for rank four scheduling. *Operations Research Letters*, 42(5):348–350, 2014.

[12] Friedrich Eisenbrand and Gennady Shmonin. Carathéodory bounds for integer cones. *Operations Research Letters*, 34(5):564–568, 2006.

[13] Jan Clemens Gehrke, Klaus Jansen, Stefan EJ Kraft, and Jakob Schikowski. A ptas for scheduling unrelated machines of few different types. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 290–301. Springer, 2016.

[14] Michel X Goemans and Thomas Rothvoß. Polynomiality for bin packing with a constant number of item types. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 830–839. Society for Industrial and Applied Mathematics, 2014.

[15] Dorit S Hochbaum and David B Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM (JACM)*, 34(1):144–162, 1987.

[16] Ellis Horowitz and Sartaj Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM (JACM)*, 23(2):317–327, 1976.

[17] Csanad Imreh. Scheduling problems on two sets of identical machines. *Computing*, 70(4):277–294, 2003.

[18] Klaus 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(2):457–485, 2010.

[19] Klaus Jansen, Kim-Manuel Klein, and José Verschae. Closing the gap for makespan scheduling via sparsification techniques. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 72:1–72:13, 2016.

[20] Klaus Jansen and Marten Maack. An EPTAS for scheduling on unrelated machines of few different types. In *Algorithms and Data Structures - 15th International Symposium, WADS 2017, St. John's, NL, Canada, July 31 - August 2, 2017, Proceedings*, pages 497–508, 2017. URL: `https://doi.org/10.1007/978-3-319-62127-2_42`, `doi:10.1007/978-3-319-62127-2_42`.

[21] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of operations research*, 12(3):415–440, 1987.

[22] Dušan Knop and Martin Koutecký. Scheduling meets n-fold integer programming. *arXiv preprint arXiv:1603.02611*, 2016.

[23] Jan Karel Lenstra, David B Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical programming*, 46(1-3):259–271, 1990.

[24] Hendrik W Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of operations research*, 8(4):538–548, 1983.

[25] Gurulingesh Raravi and Vincent Nélis. A ptas for assigning sporadic tasks on two-type heterogeneous multiprocessors. In *Real-Time Systems Symposium (RTSS), 2012 IEEE 33rd*, pages 117–126. IEEE, 2012.

[26] Andreas Wiese, Vincenzo Bonifaci, and Sanjoy Baruah. Partitioned edf scheduling on a few types of unrelated multiprocessors. *Real-Time Systems*, 49(2):219–238, 2013.

[27] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.

[28] Gerhard J Woeginger. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? *INFORMS Journal on Computing*, 12(1):57–74, 2000.