INSTITUT FÜR INFORMATIK

Scheduling jobs on uniform processors revisited

Klaus Jansen
Christina Robenek

Bericht Nr. 1109
November 2011
ISSN 2192-6247

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

ZU KIEL

Institut für Informatik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

Scheduling jobs on uniform processors revisited

Klaus Jansen
Christina Robenek

Bericht Nr. 1109
November 2011
ISSN 2192-6247

e-mail: $\{kj, cot\}$@informatik.uni-kiel.de

Parts of this results have been published as extended abstract in Proceedings of the 9th Workshop on Approximation and Online Algorithms (WAOA 2011). The research was supported by DFG project JA612/14-1, "'Design and analysis of efficient polynomial approximation schemes for scheduling and related optimization problems"'.

Abstract

We study the problem of scheduling jobs on uniform processors with the objective to minimize the makespan. In scheduling theory this problem is known as $Q||C_{\max}$. We present an EPTAS for scheduling on uniform machines avoiding the use of an MILP or ILP solver. Instead of solving (M)ILPs we solve the LP-relaxation and use structural information about the "closest" ILP solution. For a given LP-solution x we consider the distance to the closest ILP solution y in the infinity norm, i.e. $\|x-y\|_\infty$. We call this distance $\text{max-gap}(A_\delta)$, where A_δ is the constraint matrix of the considered (I)LP. For identical machines and $\delta = \Theta(\varepsilon)$ the matrix A_δ has integral entries in $\{0, \ldots, (1+\delta)/\delta\}$ and $O(1/\delta \log(1/\delta))$ rows representing job sizes and $2^{O(1/\delta \log^2(1/\delta))}$ columns representing configurations of jobs, so that the column sums are bounded by $(1+\delta)/\delta$. The running-time of our algorithm is $2^{O(1/\varepsilon \log(1/\varepsilon)\log(C(A_\delta))} + O(n \log n)$ where $C(A_\delta)$ denotes an upper bound for $\text{max-gap}(A_\delta)$. Furthermore, we can generalize the algorithm for uniform machines and obtain a running-time of $2^{O(1/\varepsilon \log(1/\varepsilon)\log(C(\tilde{A}_\delta))} + poly(n)$, where $\tilde{A}_\delta$ is the constraint matrix for a sub-problem considered in this case. In both cases we show that $C(A_\delta), C(\tilde{A}_\delta) \leq 2^{O(1/\varepsilon \log^2(1/\varepsilon))}$. Consequently, our algorithm has running-time at most $2^{O(1/\varepsilon^2 \log^3(1/\varepsilon))} + O(n \log n)$ for identical machines and $2^{O(1/\varepsilon^2 \log^3(1/\varepsilon))} + poly(n)$ for uniform machines, the same as in [12]. But, to our best knowledge, no instance is known to take on the value $2^{O(1/\varepsilon \log^2(1/\varepsilon))}$ for $\text{max-gap}(A_\delta)$ or $\text{max-gap}(\tilde{A}_\delta)$. If $C(\tilde{A}_\delta), C(A_\delta) \leq poly(1/\varepsilon)$, the running-time of the algorithm would be $2^{O(1/\varepsilon \log^2(1/\varepsilon))} + poly(n)$ and thus improve the result in [12].

1 Introduction

In this paper we study the problem of scheduling jobs on uniform processors with the objective to minimize the makespan. In scheduling theory this problem is known as $Q||C_{\max}$ and is formally described as follows. We are given a set $\mathcal{J}$ of n Jobs J_j with processing times p_j and a set $\mathcal{P}$ of m processors P_i, each of them running with a certain speed s_i. A job J_j needs p_j/s_i time units to be finished, if it is executed on P_i. Without loss of generality we assume that the number m of processors is bounded by the number n of jobs and that the processors are sorted by decreasing speed, i.e. $s_1 \geq s_2 \geq \ldots \geq s_m$. For an instance $\mathcal{I}$ let $OPT(\mathcal{I})$ denote the length of an optimum schedule

A *polynomial time approximation scheme* (PTAS) for $Q||C_{\max}$ is a family of polynomial-time approximation algorithms $(A_\varepsilon)_{\varepsilon>0}$, where for an instance $\mathcal{I}$ the output of each algorithm A_ε is a schedule of length $(1+\varepsilon)OPT(\mathcal{I})$ and the running-time of A_ε is bounded by a polynomial in the input length $|\mathcal{I}|$. The running-time of every A_ε is allowed to be exponential in $1/\varepsilon$, which can lead to very large running-times if ε is very small. Therefore we distinguish furthermore *efficient polynomial-time approximation schemes* (EPTAS) that have running-time bounded by $f(1/\varepsilon)poly(|\mathcal{I}|)$ for a function f, and *fully polynomial-time approximation schemes* (FPTAS) with running-time bounded by a polynomial in both, $1/\varepsilon$ and $|\mathcal{I}|$.

Known Results. In [5] and [6] the problem was shown to be NP-hard even for identical processors. In 1976 Horowitz and Sahni [11] presented an approximation scheme for a constant number m of uniform processors. Later Gonzales et al. [5] showed for the same problem that the LPT list algorithm (using largest processing time first policy) has output in $[1.5\ OPT(\mathcal{I}), 2\ OPT(\mathcal{I})]$. Hochbaum and Shmoys presented a PTAS for $Q||C_{\max}$ with running-time $(n/\varepsilon)^{O(1/\varepsilon^2)}$ [9],[10]. For identical processors the complexity was improved to $(n/\varepsilon)^{O(1/\varepsilon \log(1/\varepsilon))}$ by Leung [18]. Since the problem was shown to be NP-hard in the strong sense [5], no FPTAS exists. But, for identical processors Hochbaum [8] and Alon et al. [1] developed an EPTAS with running-time $f(1/\varepsilon) + O(n)$, where f is a function doubly exponential in $1/\varepsilon$. In [12] Jansen gave an EPTAS for scheduling jobs on uniform processors using an MILP relaxation with running-time $2^{O(1/\varepsilon^2 \log^3(1/\varepsilon))} + poly(n)$. Sanders et al. obtained a robust online algorithm for scheduling on identical machines with competitive ratio $(1+\varepsilon)$ and migration factor $\beta(\varepsilon) = 2^{O(1/\varepsilon \log^2(1/\varepsilon))}$ so that the running-time for incorporating a newly arrived job is constant. It maintains and updates a data structure in time doubly exponential in $1/\varepsilon$, namely $2^{2^{O(1/\varepsilon \log^2(1/\varepsilon))}}$, in each iteration. This is done by comparing the distance between solutions for ILPs with different right hand sides. The general case for uniform processors is not considered.

Our Results. In this work we present an EPTAS for scheduling on uniform machines avoiding the use of an MILP or ILP solver. In our new approach instead of solving (M)ILPs we solve the LP-relaxation and use structural information about the "closest" ILP solution. For a given LP-solution x we consider the distance to the closest ILP solution y in the infinity norm, i.e. $\|x - y\|_\infty$. For the constraint matrix A_δ of the considered LP we call this distance

$$\max\text{-gap}(A_\delta) := \max\{\min\{\|y^\star - x^\star\|_\infty : y^\star \text{solution of ILP}\} : x^\star \text{solution of LP}\}.$$

Let $C(A_\delta)$ denote an upper bound for $\max\text{-gap}(A_\delta)$. The running-time of our algorithm is $2^{O(1/\varepsilon \log(1/\varepsilon) \log(C(A_\delta))} + poly(n)$. We show that $C(A_\delta) \leq 2^{O(1/\varepsilon \log^2(1/\varepsilon))}$. Consequently, our algorithm has running-time at most $2^{O(1/\varepsilon^2 \log^3(1/\varepsilon))} + poly(n)$, the same as in [12]. But, to our best knowledge, no instance is known to take on the value $2^{O(1/\varepsilon \log^2(1/\varepsilon))}$ for $\max\text{-gap}(A_\delta)$. We conjecture $C(A_\delta) \leq poly(1/\varepsilon)$. If that holds, the running-time of the algorithm would be $2^{O(1/\varepsilon \log^2(1/\varepsilon))} + poly(n)$ and thus improve the result in [12].

Methods. Assume that we are given an instance $\bar{\mathcal{I}}$ of m identical processors and n jobs with only d different processing times p_j, such that there are n_j jobs of each size. We use the dual approximation method by Hochbaum and Shmoys [10] to find a value T for the optimum makespan and transform the scheduling problem into a bin packing problem with bin size T. Then the problem can be described via the following configuration ILP for d different item sizes:

$$\begin{aligned} \sum_i x_i &\leq m \\ \sum_i a(j,i) x_i &\geq n_j \text{ for } j = 1, \dots, d \\ x_i &\in \mathbb{Z}_{\geq 0}. \end{aligned} \tag{$ILP(d)$}$$

A configuration C_i is a multiset of processing times p_j so that their total sum is bounded by T. The integer $a(j,i)$ denotes the number of jobs of processing time p_j in C_i. In $ILP(d)$

the variable x_i is the number of bins in which jobs are packed according to configuration C_i. Solving an ILP is always difficult [15, 16], so what kind of information about the structure of the ILP-solution can we get from a solution of the LP-relaxation?
For the constraint matrix $A := (a(j,i))_{ji}$ of the above $ILP(d)$ we consider max-gap(A). Having an upper bound $C(A)$ for max-gap(A) and having an optimum fractional solution $x^\star$ we conclude that there exists an optimum solution $y^\star$ of $ILP(d)$ so that $y_i^\star \geq \lceil x_i^\star - C(A) \rceil$ for $x_i^\star \geq C(A)$. So we know how a subset of the bins $\mathcal{B}' \subset \mathcal{B}$ has to be filled with jobs in the optimum solution $y^\star$. We can reduce the instance to an instance $\bar{\mathcal{I}}_{red}$ by taking out the bins in $\mathcal{B}'$ and those jobs that are packed in $\mathcal{B}'$:

$$
\begin{aligned}
\tilde{m} &:= m - \sum_{x_i^\star > C(A)} \lceil x_i^\star - C(A) \rceil \text{ processors} \\
\tilde{n}_j &:= n_j - \sum_{x_i^\star > C(A)} a(j,i) \lceil x_i^\star - C(A) \rceil \text{ for all processing times } p_j.
\end{aligned}
\tag{1}
$$

In Figure 1 for example we have $C(A) = 3$. Given an optimum fractional solution $x^\star$ we conclude that there exists an optimum solution $y^\star$ of the ILP with $\|x^\star - y^\star\|_\infty \leq 3$. Thus, if $x_i^\star = 7.5$ we have $y_i^\star \geq 5$. Therefore we know that there is an integral solution of ILP (1) where at least 5 bins are occupied with configuration C_i. We take out these 5 bins and the corresponding jobs. Keep in mind that if the number of different

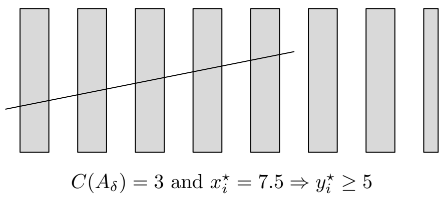

Figure 1: Reducing the instance.

job sizes and the number of jobs per bin is bounded by a constant, the total number of remaining jobs in $\bar{\mathcal{I}}_{red}$ can be bounded by a function in the value $C(A)$ namely by #(non-zero variables in LP-solution) $*$ #(jobs per bin) $* \, C(A)$. Cook et al. [3] showed for general (I)LPs that max-gap(A) is bounded by Δ times the number of variables, where Δ is the maximum absolute value of a subdeterminant of the constraint matrix A.

So for an instance $\mathcal{I}$ with identical machines our algorithm (for a formal description see Algorithm 1) first chooses $\delta \in \Theta(\varepsilon)$ and finds by binary search a candidate T for the makespan with $OPT(\mathcal{I}) \leq T \leq (1+\delta)OPT(\mathcal{I})$. By scaling we can assume that $T = 1$ and round the processing times p_j to values $\bar{p}_j = \delta(1+\delta)^{k_j}$ with $k_j \in \mathbb{Z}$ such that $p_j \leq \bar{p}_j \leq (1+\delta)p_j$.

Consequently, we have to enlarge the bin capacities slightly to $(1+\delta)T = (1+\delta)$. With $\bar{\mathcal{I}}$ we denote the instance of rounded jobs, that are large, i.e. $\bar{p}_j > \delta$. Notice that we have at most $O(1/\delta)$ large jobs per bin and by the rounding we have $R \in O(1/\delta \log(1/\delta))$ different large job sizes (see also Lemma 2.1). We set up a configuration ILP for $\bar{\mathcal{I}}$ with $2^{O(1/\delta \log^2(1/\delta))}$ variables and constraint matrix A_δ as described above and solve the LP-relaxation or decide that no solution exists. In the latter case we increase the value T

and restart. Solving the LP-relaxation can be done in time $poly(1/\delta, \log n)$ [7]. Using the theorem by Cook et al. [3] we show that $C(A_\delta)$ is at most $2^{O(1/\delta \log^2(1/\delta))}$ (refer to Lemma 2.3). Having a solution of the LP-relaxation we can reduce the instance as described in equation (1). Since the number of non-zero variables in a basic solution of the LP-relaxation is bounded by $rank(A_\delta) \in O(1/\delta \log(1/\delta))$, the number of remaining large jobs in $\bar{\mathcal{I}}_{red}$ is bounded by $2^{O(1/\delta \log(1/\delta) \log(C(A_\delta)/\delta))}$. We allocate them by a dynamic programming approach. If this fails, we increase T and restart. In the end the small jobs are added greedily. The running-time is composed as follows "sorting the items by size"+"binary search on T"∗"solving the LP"∗"dynamic program"+"adding small jobs". This gives total running-time $O(n \log n) + O(\log(1/\varepsilon)) poly(1/\varepsilon, \log n) 2^{O(1/\varepsilon \log(1/\varepsilon) \log(C(A_\delta)))} + O(n)$ $\leq O(n \log n) + poly(\log n) 2^{O(1/\varepsilon \log(1/\varepsilon) \log(C(A_\delta)))} \leq 2^{O(1/\varepsilon \log(1/\varepsilon) \log(C(A_\delta)))} + O(n \log n)$.

The algorithm for uniform processors is much more complex. Since we have different bin capacities for uniform machines, we cannot directly apply the techniques used for identical machines. Therefore, we distinguish between three different scenarios for the shape of the bin sizes. For each scenario we give an algorithm that solves the problem in time $2^{O(1/\delta \log(1/\delta) \log(C(\tilde{A}_\delta)))} + poly(n)$ that applies our new technique to a subset of the instance. Furthermore, we use a new technique to round LP solutions for fractional bin packing producing only few extra bins. In all cases the running time depends on $C(\tilde{A}_\delta)$ and is $2^{O(1/\varepsilon^2 \log^3(1/\varepsilon))} + poly(n)$ in the worst case for $C(\tilde{A}_\delta)$. Here, the matrix $\tilde{A}_\delta$ describes the constraints appearing in an ILP-approach that characterizes a more general scheduling problem as the one for identical machines: the jobs have a bounded number of different sizes and the machines run group-wise with the same speed, so we have configurations for each group. The entries of $\tilde{A}_\delta$ are integers in $\{0, 1, \ldots, (1+\delta)g(1/\delta)/\delta\}$ and the column sums are bounded by $(1+\delta)g(1/\delta)/\delta + 1$ for a function $g(1/\delta) = poly(1/\delta)$ that will be specified later. The value $C(\tilde{A}_\delta)$ is an upper bound for max-gap$(\tilde{A}_\delta)$.
We found out that the value Δ for matrices describing scheduling problems can be exponential in the number of different item sizes (see Lemma 2.5). But, no instance is known to take on the upper bound for max-gap(A_δ) or max-gap$(\tilde{A}_\delta)$. Therefore, an open question is to find a better bound for max-gap(A_δ) and max-gap$(\tilde{A}_\delta)$. One can also think of a robust online algorithm for identical processors or even for uniform processors with improved running-time using similar techniques.

Organization of the paper. In Section 2 we describe the algorithm for identical processors fully detailed. Moreover, we show that max-gap(A_δ) is bounded from above and that the value Δ is bounded from below. In Section 3 we present an efficient algorithm for uniform processors that avoids to solve (M)ILPs and uses an upper bound for max-gap$(\tilde{A}_\delta)$ instead. Here, we proceed by case distinction and consider three different scenarios for the bin sizes. Consequently, a subdivision of Section 3 is given by the three cases.

2 Scheduling on Identical Processors

An overview about our algorithm is given in Algorithm 1. Below we describe our approach step by step.

Algorithm 1 Algorithm for identical machines

1: Obtain with list scheduling algorithm a schedule of length $LS(\mathcal{I}) \geq 2OPT(\mathcal{I})$.
2: Compute a value $T \in [LS(\mathcal{I})/2, LS(\mathcal{I})]$.
3: Round the processing times of the jobs and distinguish small and large jobs
4: Solve an LP-relaxation and find a schedule for a subset of the large jobs and reduce the instance to $\mathcal{I}_{red}$.
5: **if** LP has no feasible solution **then**
6:     increase T and go to step 3 .
7: **end if**
8: Dynamic program for $\mathcal{I}_{red}$.
9: **if** the dynamic program for $\mathcal{I}_{red}$ does not find a feasible solution for m machines **then**
10:     increase T and go to step 3.
11: **end if**
12: Schedule the small jobs behind the large ones. .

Computing an approximate value for the makespan. We first compute a near optimum value T for the makespan. Let $LS(\mathcal{I})$ be the length of a schedule generated by the list scheduling algorithm with $LS(\mathcal{I}) \leq 2OPT(\mathcal{I})$ where $OPT(\mathcal{I})$ is the minimum schedule length among all solutions. This implies that $LS(\mathcal{I})/2 \leq OPT(\mathcal{I}) \leq LS(\mathcal{I})$. In the following we choose a value $\delta \leq \min(\varepsilon/4, 1/2)$. Using binary search over the interval $[LS(\mathcal{I})/2, LS(\mathcal{I})]$ we compute a value T such that $OPT(\mathcal{I}) \leq T \leq (1+\delta)OPT(\mathcal{I})$. Notice that the interval $[LS(\mathcal{I})/2, LS(\mathcal{I})]$ can be divided into $1/\delta$ subintervals of length $(\delta/2)LS(\mathcal{I}) \leq \delta OPT(\mathcal{I})$.
To find the corresponding interval, we use the dual approximation method by Hochbaum and Shmoys [9] that for a given value T either computes an approximate schedule of length at most $T(1+\delta)$ or shows that there is no schedule of length T. Using binary search over the interval, we compute a value $T \leq (1+\delta)OPT(\mathcal{I})$ and an approximate schedule below of length at most $(1+2\delta)T \leq (1+2\delta)(1+\delta)OPT(\mathcal{I}) \leq (1+4\delta)OPT(\mathcal{I}) \leq (1+\varepsilon)OPT(\mathcal{I})$ for $\delta \leq 1/2$ and $\delta \leq \varepsilon/4$. Notice that $O(\log(1/\varepsilon))$ steps are sufficient in the binary search.

Rounding. As described in the introduction the scheduling problem can be transformed into a bin packing problem with bin sizes T [2]. By scaling we may assume that $T = 1$. Let $\bar{\mathcal{I}}$ be the instance with the rounded processing times. Clearly, for any set A of jobs with $\sum_{j \in A} p_j \leq 1$ we have for the total rounded processing time $\sum_{j \in A} \bar{p}_j \leq (1+\delta)$. We partition the jobs now into large and small jobs. A job j is large, if the processing time $\bar{p}_j \geq \delta$. Other jobs with $\bar{p}_j < \delta$ are called small.

Lemma 2.1. *Let A be a set $\{a(1+\delta)^x, \ldots, a(1+\delta)^y\}$ with $x, y \in \mathbb{Z}^+$, $x < y$ and $a \in \mathbb{R}^+$. Then $|A| \geq \log(\max(A)/\min(A))/\delta + 1$ and $|A| \leq 2\log(\max(A)/\min(A))/\delta + 1$ for any $\delta \in (0, 1/2]$.*

Proof. Using the assumption on A, $\max(A)/\min(A) = (1+\delta)^{y-x}$. Therefore, the number of elements $a(1+\delta)^i$ in A is equal to

$$y - x + 1 = \log(\max(A)/\min(A))/\log(1+\delta) + 1.$$

Using $\log(1+\delta) \geq \delta - \delta^2 \geq \delta/2$ for $\delta \in (0,1/2]$ and $\log(1+\delta) \leq \delta$, the cardinality of A is at least $\log(\max(A)/\min(A))/\delta + 1$ and at most $2\log(\max(A)/\min(A))/\delta + 1$. $\square$

By the lemma above the number of different large rounded processing times is bounded by $R = 2\log((1+\delta)/\delta)/\delta + 1 = O(1/\delta \log(1/\delta))$. With n_j we denote the number of jobs in $\bar{\mathcal{I}}$ with processing time $\delta(1+\delta)^j$ for $j = 0, \dots, R-1$.

(I)LP-formulation for large jobs The assignment of the large jobs to the bins can be formulated as a configuration ILP as in $ILP(d)$ for $d = R$ with exponentially many variables. We denote it with $ILP(R)$. Here a configuration is a multiset of large rounded processing times, i.e. values $\delta(1+\delta)^j \in [\delta, (1+\delta)]$ so that their total sum is bounded by $1+\delta$. Since there are at most $(1+\delta)/\delta$ many large jobs on a processor with makespan $(1+\delta)$, the number of multisets with large processing times is bounded by $((1+\delta)/\delta)^{O(1/\delta \log(1/\delta))} = 2^{O(1/\delta \log^2(1/\delta))}$. Let $a(j,i)$ be the number of occurrences of value $\delta(1+\delta)^j$ in C_i. Consequently the matrix $A_\delta := (a(j,i))_{ji}$ is the constraint matrix of $ILP(R)$. To avoid now the algorithm by Lenstra or Kannan to compute an optimum solution of $ILP(R)$, we consider its LP-relaxation $LP(R)$. We can solve $LP(R)$ using Theorem (6.6.5) in [7]:
The first constraint of $LP(R)$ can be transformed into an objective function $\min cx$ with $c \equiv 1$. The resulting dual is a maximization problem of the form $\max \left\{ \sum_{j=0}^{R-1} n_j y_j \middle|\, y \in P \right\}$, where $P = \left\{ y \in \mathbb{R}^R \middle|\, \sum_j a(j,i) y_j \leq 1, 1 \leq i \leq C, y \geq 0 \right\}$. The polyhedron P has facet complexity at most $\varphi := R(\lceil \log(\frac{1+\delta}{\delta}) \rceil + 1) + 2 = O(1/\delta \log^2(1/\delta))$, since each inequality describing P has encoding length at most φ. According to Theorem (6.6.5) in [7] there is an algorithm that finds a basic optimum solution of the dual of the dual, i.e. the modified $LP(R)$. The running-time is bounded by a polynomial in φ and in the encoding length of $(n_0, \dots, n_{R-1})$, which is in $O(1/\delta \log(1/\delta) \log(n))$. Thus, the running-time is bounded by $poly(1/\delta, \log(n))$.

Reducing the instance of large jobs. Let $x^\star$ be an optimum basic solution of $LP(R)$ and let $C(A_\delta)$ be an upper bound for max-gap(A_δ). Let $\bar{\mathcal{I}}_{red}$ be the corresponding reduced instance achieved as in (1). From $x^\star$ we obtain a packing for the large rectangles in $\bar{\mathcal{I}} \backslash \bar{\mathcal{I}}_{red}$ as described in the introduction. Now, each basic solution of the LP-relaxation has at most $O(1/\delta \log(1/\delta))$ many variables with value larger than 0. In addition the reduced variables $x_i - \lceil x_i^* - C(A_\delta) \rceil$ are bounded by $C(A_\delta)$. Since $\sum_j a(j,i) \leq (1+\delta)/\delta$ for each configuration C_i, we can bound the number of remaining jobs in $\bar{\mathcal{I}}_{red}$, i.e. $\tilde{n} \leq O(1/\delta^2 \log(1/\delta) C(A_\delta))$. Furthermore, we may suppose that $\tilde{m} \leq \tilde{n}$; since more processors are not necessary.

Dynamic program for reduced instance. An assignment of the remaining jobs in $\mathcal{I}_{red}$ can be computed via dynamic programming in time

$$(\tilde{n})^{O(1/\delta \log(1/\delta))} \leq 2^{O(1/\delta \log(1/\delta) \log(C(A_\delta)/\delta))}.$$

We simply go over the machines and store vectors $(x_1, \dots, x_R)$ with numbers x_j of jobs with processing time $\bar{p}_j$ for $j = 1, \dots, R$ used by the first k processors for $k = 1, \dots, \tilde{m}$. If the algorithm does not find a feasible assignment, we know that the original $ILP(R)$ has no feasible solution and have to increase T.

Small jobs. Once we have an assignment of the large jobs, we can pack the small jobs. They can be packed greedily onto the processors, if the total size of the small and large jobs is bounded by m (the total sum of all bin capacities). Adding the small jobs increases the makespan by at most δ. Therefore, the overall makespan is at most $(1+2\delta)$. If the condition above does not hold or if $p_{max} = \max_{j=1,\dots,n} p_j > 1$, there is no schedule of length $T = 1$.

2.1 Bounds for $\max$-gap(A_δ) and the running-time

To obtain an upper bound $C(A_\delta)$ for max-gap(A_δ) we use an interesting result by Cook et al. [3]. They proved that the maximum distance between an optimum solution of the LP and a closest optimum solution of the ILP (and vice versa) is bounded by a function in the dimension and the coefficients of the underlying matrix.

Theorem 2.2. *[3] Let A be an integral $(M \times N)$ matrix, such that each subdeterminant is at most Δ in absolute value, and let b and c be vectors. Suppose that both objective values (i) $\min\{c^T x | Ax \geq b\}$ and (ii) $\min\{c^T x | Ax \geq b;\ x \in \mathbb{Z}^N\}$ are finite. Then:*

(a) for each optimum solution y of (i) there exists an optimum solution z of (ii) with $\|y - z\|_\infty \leq N\Delta$ and

(b) for each optimum solution z of (ii) there exists an optimum solution y of (i) with $\|y - z\|_\infty \leq N\Delta$.

Note that the theorem above also holds, if we have additional inequalities of the form $x_i \geq 0$. Furthermore, we can use $c^T x = \sum_i x_i$ as objective function instead of the inequality $\sum_i x_i \leq m$ in $ILP(d)$. For scheduling on identical processors the objective values of the ILP formulation for the rounded large jobs $\bar{\mathcal{I}}$ and its LP relaxation both are finite. Consequently, max-gap(A_δ) is bounded by $N\Delta$. In the following we give bounds for the parameters N and Δ.

Lemma 2.3. *The number of variables N in the modified ILP, the maximum absolute value Δ over all subdeterminants corresponding to the matrix A_δ and* max-gap(A_δ) *are at most $2^{O(1/\delta \log^2(1/\delta))}$.*

Proof. The number N of variables in the modified ILP is equal to the number of configurations, so it is bounded by $2^{O(1/\delta \log^2(1/\delta))}$. On the other hand, the absolute value of the determinant of a quadratic $(M \times M)$ sub-matrix A of A_δ with column vectors $A_1, \dots, A_M$ and $|\det(A)| = \Delta$ can be bounded using the Hadamard's inequality

$$|\det(A)| \leq \prod_{i=1}^{M} \|A_i\|_2. \tag{2}$$

Since the coefficients of a column A_i correspond to a configuration C_i, the sum of the entries is bounded by $(1+\delta)/\delta$. Therefore, using the inequality $\|A_i\|_2 \leq \|A_i\|_1$, we obtain

$$\|A_i\|_2 \leq \|A_i\|_1 \leq \sum_{j=1}^{O(1/\delta \log(1/\delta))} a(j,i) \leq (1+\delta)/\delta. \tag{3}$$

This implies that Δ is at most

$$\Delta \overset{(2)}{\leq} \prod_{i=1}^{O(1/\delta \log(1/\delta))} \|A_i\|_2 \overset{(3)}{\leq} \prod_{i=1}^{O(1/\delta \log(1/\delta))} \frac{(1+\delta)}{\delta} = 2^{O(1/\delta \log^2(1/\delta))}. \quad (4)$$

We conclude that max-gap(A_δ) is also at most $2^{O(1/\delta \log^2(1/\delta))}$. □

This implies the following theorem

Theorem 2.4. *For a list of n jobs $\mathcal{I}$ Algorithm 1 produces a schedule on m machines with makespan at most $(1+\varepsilon)OPT(\mathcal{I})$ in time*

$$2^{O(1/\varepsilon \log(1/\varepsilon)\log(C(A_\delta)))} + O(n \log n) \leq 2^{O(1/\varepsilon^2 \log^3(1/\varepsilon))} + O(n \log n).$$

If $C(A_\delta) = poly(1/\varepsilon)$, the running-time improves to $2^{O(1/\varepsilon \log^2(1/\varepsilon))} + O(n \log n)$.

On the other hand, the value Δ can be quite large. But note that a lower bound for max-gap(A_δ) remains unknown.

Lemma 2.5. *The maximum value Δ over all subdeterminants of the coefficient matrix $A_\delta = (a(j,i))$ is at least $2^{\Omega(1/\delta \log^2(1/\delta))}$.*

Proof. The number of rounded processing times $\bar{p}_j$ in the interval $[\delta, \delta^{1/2}]$ is at least $\log(\delta^{1/2}/\delta)/\delta = 1/\delta \log(\delta^{-1/2})$. For each rounded processing time $\bar{p}_j = \delta(1+\delta)^j \in [\delta, \delta^{1/2}]$ for $j = 0, \ldots, 1/\delta \log(\delta^{-1/2}) - 1$, we take one configuration or multiset C_j which consists of only $\delta^{-1/2}$ numbers $\bar{p}_j$, i.e.

$$C_j = \{0 : \delta(1+\delta)^0, \ldots, 0 : \delta(1+\delta)^{j-1}, \delta^{-1/2} : \delta(1+\delta)^j,$$
$$0 : \delta(1+\delta)^{j+1}, \ldots, 0 : \delta(1+\delta)^{1/\delta \log(\delta^{-1/2})-1}\}.$$

The determinant of the matrix corresponding to these configurations
$C_0, \ldots, C_{1/\delta \log(1/\delta^{1/2})-1}$ is

$$(\delta^{-1/2})^{\Omega(1/\delta \log(\delta^{-1/2}))} = 2^{\Omega(1/\delta \log^2(\delta^{-1/2}))} = 2^{\Omega(1/\delta \log^2(1/\delta))}.$$

This implies that $\Delta \geq 2^{\Omega(1/\delta \log^2(1/\delta))}$. □

3 Scheduling on Uniform Processors

For uniform processors we can compute a 2 - approximation using the LPT algorithm studied by Gonzales et al. [5]. Here $LPT(\mathcal{I}) \leq 2OPT(\mathcal{I})$ where $LPT(\mathcal{I})$ is the schedule length generated by the LPT algorithm. Similar to identical processors, we can split the interval $[LPT(\mathcal{I})/2, LPT(\mathcal{I})]$ into $1/\delta$ subintervals of length $(\delta/2)LPT(\mathcal{I}) \leq \delta OPT(\mathcal{I})$

and transform the scheduling problem with makespan T into a bin packing problem with bin sizes $c_1 \geq \ldots \geq c_m$ (where $c_i = T \cdot s_i$). By scaling we assume $c_m = 1$. As for identical machines we round the job sizes p_j to values $\bar{p}_j = \delta(1+\delta)^{k_j} \leq (1+\delta)p_j$. Additionally we round and increase slightly the bin capacities c_i to values $\bar{c}_i = (1+\delta)^{\ell_i} \leq c_i(1+\delta)^2$. Let the instance of rounded jobs and bin capacities be denoted with $\bar{\mathcal{I}}$. For a set of bins $\mathcal{B}$ let $c_{\min}(\mathcal{B}) := \min\{c_i | b_i \in \mathcal{B}\}$. Analogously we define $c_{\max}(\mathcal{B})$.

Lemma 3.1. *[12] If there is a feasible packing of n jobs with processing times p_j into m bins with capacities $c_1, \ldots, c_m$, then there is also a packing of the n jobs with rounded processing times $\bar{p}_j = \delta(1+\delta)^{k_j} \leq (1+\delta)p_j$ into the m bins with rounded bin capacities $\bar{c}_i = (1+\delta)^{\ell_i} \leq c_i(1+\delta)^2$.*

In the general case with different bin sizes, we distinguish between three different scenarios depending on the structure of the set of bins in the instance. Let $g : \mathbb{N} \longrightarrow \mathbb{N}$ and $f : \mathbb{N} \longrightarrow \mathbb{N}$ be functions so that $g(1/\delta) \geq 1/\delta \log^2(1/\delta)$ with $g = poly(1/\delta)$ and $f(1/\delta) = \max\{\lceil (1+\delta+\log^2(1/\delta))/\delta \rceil, 1/\delta^4 \log(g(1/\delta)/\delta) C(\tilde{A}_\delta)\}$. The constant $C(\tilde{A}_\delta)$ is still an upper bound for max-gap$(\tilde{A}_\delta)$. Here $\tilde{A}_\delta$ is a matrix corresponding to a more general scheduling problem with $O(1/\delta \log(1/\delta))$ rows (different job sizes) and $2^{O(1/\delta \log^2(1/\delta))}$ columns (configurations) with integral entries in $\{0, 1, \ldots, (1+\delta)g(1/\delta)/\delta\}$ and column sums bounded by $(1+\delta)g(1/\delta)/\delta + 1$ similar to the constraint matrix of the configuration ILP used for identical processors. We consider the following three scenarios:

Case 1: For all $i \in \{1, \ldots, m\}$ we have $\bar{c}_1/\bar{c}_m \leq g(1/\delta)$.
Case 2: There exists an index $K+1 \leq f(1/\delta)$ with $\bar{c}_1/\bar{c}_i < g(1/\delta)$ for $1 \leq i \leq K$ and $\bar{c}_1/\bar{c}_i \geq g(1/\delta)$ for $K+1 \leq i \leq m$.
Case 3: There exists an index $K+1 > f(1/\delta)$ with $\bar{c}_1/\bar{c}_i < g(1/\delta)$ for $1 \leq i \leq K$ and $\bar{c}_1/\bar{c}_i \geq g(1/\delta)$ for $K+1 \leq i \leq m$.

In the first scenario all bins have similar capacities. More precisely the capacity of every bin is bounded from above by $g(1/\delta)$ (Keep in mind that $c_{\min} := \min_i c_i = 1$). This scenario can be solved very similar to the problem with identical machines.
In the second scenario we consider two different bin groups $\mathcal{B}_0 := \{b_1, \ldots, b_K\}$ and $\mathcal{B}_1 := \{b_{K+1}, \ldots, b_m\}$. For $\mathcal{B}_0$ we preprocess an assignment of large jobs ($\bar{p}_j > \delta c_{\min}(\mathcal{B}_0)$) via a dynamic program and obtain a set of assignment vectors V. If the dynamic program does not find a feasible solution for $\mathcal{B}_0$, we increase T. For $v \in V$ we allocate large jobs fractionally into $\mathcal{B}_1$ via an LP. If the LP does not have a feasible solution we compute a different vector v. If we still do not find an LP solution, we increase T. Then we round the solution of the LP with a novel rounding technique using a subroutine for bin packing with different bin sizes that produces only few extra bins. In the end the small jobs are scheduled behind the large ones. An overview of the overall algorithm gives Algorithm 2.
The third scenario is the most complicated case. Here we have three bin groups $\mathcal{B}_0 = \{b_1, \ldots, b_K\}$, $\mathcal{B}_1 = \{b_i | i > K, \bar{c}_i \geq \delta c_{\min}(\mathcal{B}_0)\}$ and the remaining bins $\mathcal{B}_2 = \mathcal{B} \setminus (\mathcal{B}_0 \cup \mathcal{B}_1)$. If $\mathcal{B}_1 \neq \emptyset$ we distinguish large, medium and small jobs, else we only have large and small jobs:
A job is called large if $\bar{p}_j > \delta c_{\min}(\mathcal{B}_0)$ and medium if $\bar{p}_j \in (\delta c_{\min}(\mathcal{B}_1), \delta c_{\min}(\mathcal{B}_0)]$; other jobs are called small. We first allocate a subset of the large jobs into $\mathcal{B}_0$ and $\mathcal{B}_1$ using a linear program. As in the case for identical machines for a given solution x of the LP we reduce the instance by the number of large jobs surely packed in the closest integral solution. If the LP has no feasible solution we have to increase T and restart. Via dynamic programming our algorithm obtains an assignment of the remaining large jobs into $\mathcal{B}_0$ and $\mathcal{B}_1$ (if there is none, increase T). The medium jobs are packed with a bin packing subroutine into $\mathcal{B}_1$. Finally, the allocated medium and large jobs are fit together with the remaining jobs and the small jobs are added. An overview of the algorithm for

this case is given in Figure 3. In the next following we describe the algorithm in each case fully detailed.

3.1 Algorithm for Case 1

Using Lemma 2.1, the number of different rounded bin sizes is at most $L = 2/\delta \log(g(1/\delta))+1$, since $g(1/\delta) = poly(1/\delta)$ we have $L = O(1/\delta \log(1/\delta))$. In this scenario a job is called *large*, if $\bar{p}_j = \delta(1+\delta)^{k_j} \geq \delta\bar{c}_m = \delta$, otherwise a it is called *small*. Let $\bar{p}_{\min}$ be the minimum large processing time, i.e. $\bar{p}_{\min} = \min\{p_j | p_j \geq \delta, j = 1, \ldots, n\}$. Clearly, $\bar{p}_{\min} \geq \delta$ if the set is non-empty. Since $\bar{p}_{\max} = \max_{j=1,\ldots,n} \bar{p}_j \leq \bar{c}_1 \leq g(1/\delta)$, we obtain $\bar{p}_{\max}/\bar{p}_{\min} \leq g(1/\delta)/\delta$. Thus, the total number of different rounded large processing times is at most $R = 2/\delta \log(g(1/\delta)/\delta) + 1$. Since $g(1/\delta) = poly(1/\delta)$ we obtain $R = O(1/\delta \log(1/\delta))$. Let n_j be the number of jobs with processing time $\delta(1+\delta)^j$ for $j = 0, \ldots, R-1$ and let m_ℓ be the number of bins with capacity or bin size $\bar{c}_\ell = (1+\delta)^{\ell-1}$ for $\ell = 1, \ldots, L$. All bins with capacity $(1+\delta)^{\ell-1}$ form a block B_ℓ of identical bins. The overall algorithm works very similar to the one in the identical case.

Allocating large jobs. For the the assignment of the large jobs to the bins, we set up an integer linear program, which is block by block similar as the one for the case of identical machines. Here we introduce for each bin size $\ell \in \{1, \ldots, L\}$ configurations $C_i^\ell \in \mathcal{C}^\ell$ as multisets of numbers $\delta(1+\delta)^j \in [\delta, \bar{c}_\ell]$, where the total sum is bounded by $\bar{c}_\ell$ for $\ell = 1, \ldots, L$. Furthermore, let $a(j, i^{(\ell)})$ be the number of occurrences of processing time $\delta(1+\delta)^j$ in configuration $C_i^{(\ell)}$. In the ILP below we use an integral variable $x_i^{(\ell)}$ to indicate the number of configurations $C_i^{(\ell)}$ in block B_ℓ.

$$\begin{aligned}
\sum_i x_i^{(\ell)} \leq m_\ell \text{ for } \quad & \ell = 1, \ldots, L \\
\sum_{i,\ell} a(j, i^{(\ell)}) x_i^{(\ell)} \geq n_j \text{ for } \quad & j = 0, \ldots, R-1 \\
x_i^{(\ell)} \in \mathbb{Z}_{\geq 0} \text{ for } \quad & i = 1, \ldots, |\mathcal{C}^{(\ell)}|, \ell = 1, \ldots, L
\end{aligned}$$

The number of constraints of the ILP (not counting the non-negativity constraints $x_i^{(\ell)} \geq 0$) is at most $R + L \leq O(1/\delta \log(1/\delta))$ and the number of variables is bounded by $(g(1/\delta)/\delta)^{O(1/\delta \log(1/\delta))} \leq 2^{O(1/\delta \log^2(1/\delta))}$, since $g(1/\delta) = poly(1/\delta)$.

To apply Theorem 2.2, we multiply the first L inequalities by (-1) and obtain $\sum_i (-1) x_i^{(\ell)} \geq -m_\ell$ for $\ell = 1, \ldots, L$. Let $\tilde{A}_\delta$ be the matrix with column vectors $A_i^{(\ell)}$ corresponding to variables $x_i^{(\ell)}$, i.e. $A_i^{(\ell)} = (0, \ldots, 0, -1, 0, \ldots, 0, a(1, i^{(\ell)}), \ldots, a(R, i^{(\ell)}))^T$. Again, let $C(\tilde{A}_\delta)$ be an upper bound for max-gap$(\tilde{A}_\delta)$. Suppose that there is feasible solution of the ILP. Our algorithm first solves the corresponding LP relaxation similar as for identical machines:
Since $g(1/\delta) = poly(1/\delta)$ the polyhedron corresponding to the dual has facet complexity bounded by $O(1/\delta \log^2(1/\delta))$ and the price vector of the dual has encoding length at most $O(1/\delta \log(1/\delta) \log(n))$. Thus, according to Theorem 6.6.5 in [7] we find a basic optimum

solution of the LP relaxation in time $poly(1/\delta, \log(n))$.

Since the ILP is feasible, the LP relaxation has a feasible solution $(\bar{x}_i^{(\ell)})$, too. Let $(y_i^{(\ell)})$ be a feasible solution of the ILP with distance (in the maximum norm) bounded by $C(\tilde{A}_\delta)$, i.e. $|y_i^{(\ell)} - \bar{x}_i^{(\ell)}| \le C(\tilde{A}_\delta)$ for all i. Such a solution y exists since both, LP and ILP are feasible. If $\bar{x}_i^\ell > C(\tilde{A}_\delta)$ we set $y_i^{(\ell)} = \lceil \bar{x}_i^{(\ell)} - C(\tilde{A}_\delta) \rceil$ and use $y_i^{(\ell)}$ configurations of type $C_i^{(\ell)}$ for block B_ℓ. Now we can reduce the instance as follows:

$$\tilde{m}_\ell := m_\ell - \sum_{i:\bar{x}_i^{(\ell)} > C(\tilde{A}_\delta)} \lceil \bar{x}_i^{(\ell)} - C(\tilde{A}_\delta) \rceil \text{ bins in block } B_\ell \text{ for } \ell = 1, \dots, L$$
$$\tilde{n}_j := n_j - \sum_{i,\ell:\bar{x}_i^{(\ell)} > C(\tilde{A}_\delta)} a(j, i^{(\ell)}) \lceil \bar{x}_i^{(\ell)} - C(\tilde{A}_\delta) \rceil \text{ jobs with rounded processing times}$$
$$\delta(1+\delta)^j \text{ for } j = 0, \dots, R-1.$$

Let $\mathcal{I}_{red}$ be the reduced instance. Notice that each basic feasible LP solution $(\bar{x}_i^{(\ell)})$ has at most $O(1/\delta \log(1/\delta))$ variables with value strictly larger than 0. Since $\bar{c}_{\max} \le g(1/\delta)$ the coefficients of the constraint matrix of the ILP $a(j, i^{(\ell)})$ are bounded by $g(1/\delta)/\delta$. By construction the reduced variables $\bar{x}_i^{(\ell)} - \lceil \bar{x}_i^{(\ell)} - C(\tilde{A}_\delta) \rceil$ in the LP are bounded by $C(\tilde{A}_\delta)$. Thus, the total number $\tilde{n} = \sum_j \tilde{n}_j$ of remaining jobs is at most $O((C(\tilde{A}_\delta) g(1/\delta))/\delta^2 \log(1/\delta))$. Furthermore, an assignment of the remaining large jobs can be computed via a dynamic program in time

$$(\tilde{n})^{O(1/\delta \log(1/\delta))} \le 2^{O(1/\delta \log(1/\delta) \log\left(\frac{C(\tilde{A}_\delta) g(1/\delta) \log(1/\delta)}{\delta^2}\right))}.$$

If the algorithm does not find a feasible assignment, then we know that the original ILP has no feasible solution.

Upper bound for $\max\text{-}\mathbf{gap}(\tilde{A}_\delta)$**.** Here we can use again Theorem 2.2 to achieve an upper bound for $\max\text{-gap}(\tilde{A}_\delta)$.

Lemma 3.2. *The maximum absolute value* Δ *of a subdeterminant of matrix* $\tilde{A}_\delta$ *and* $\max\text{-}gap(\tilde{A}_\delta)$ *are bounded by* $2^{O(1/\delta \log^2(1/\delta))}$.

Proof. Using $\|A_i^{(\ell)}\|_1 = \sum_{j=1}^R a(j, i^{(\ell)}) + 1 \le g(\frac{1}{\delta})/\delta + 1$ we obtain $\|A_i^{(\ell)}\|_2 \le g(\frac{1}{\delta})/\delta + 1$ and with Hadamard-inequality $\Delta \le (g(\frac{1}{\delta})/\delta + 1)^{O(1/\delta \log(1/\delta))} \le 2^{O(1/\delta \log^2(1/\delta))}$. Since the number of variables $N \le 2^{O(1/\delta \log^2(1/\delta))}$ and the maximum distance $\max\text{-gap}(\tilde{A}_\delta)$ can be bounded by $N\Delta$ the assertion follows with Theorem 2.2. □

Again $\Delta \ge 2^{\Omega(1/\delta \log^2(1/\delta))}$ and still a lower bound for $\max\text{-gap}(\tilde{A}_\delta)$ is open. The upper bound implies that the running-time of the dynamic program is in $2^{O(1/\delta^2 \log^3(1/\delta))}$, moreover this implies the following theorem

Theorem 3.3. *For the scenario in case 1 there is an algorithm that schedules n jobs on m uniform processors producing a schedule with makespan at most $(1+\varepsilon)OPT(\mathcal{I})$ in time*

$$2^{O(1/\varepsilon^2 \log^3(1/\varepsilon))} poly(\log(n)) + O(n \log(n)).$$

If $C(A) = poly(1/\varepsilon)$ the running-time improves to

$$2^{O(1/\varepsilon \log^2(1/\varepsilon))} poly(\log(n)) + O(n \log(n)).$$

3.2 Algorithm for Case 2

We divide the bins into two groups $\mathcal{B}_0 := \{b_1, \ldots, b_K\}$ and $\mathcal{B}_1 := \{b_{K+1}, \ldots, b_m\}$. There is a gap $g(1/\delta)$ between the largest bin b_1 and bin b_{K+1}, compare to Figure 2. In particular we have $\bar{c}_1/\bar{c}_i < g(1/\delta)$ for $b_i \in \mathcal{B}_0$ and $\bar{c}_1/\bar{c}_i \geq g(1/\delta)$ for $b_i \in \mathcal{B}_1$. A job is called *large*, if $\bar{p}_j \geq \delta c_{\min}(\mathcal{B}_0)$, other jobs are *small*.

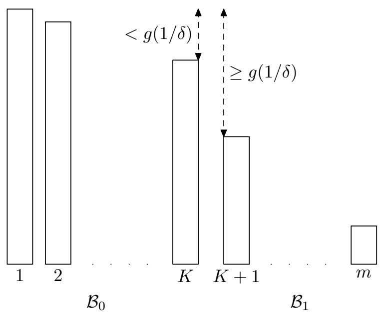

Figure 2: Shape of bins in case 2.

Preprocessing large jobs for $\mathcal{B}_0$. We first compute the set of large jobs scheduled in group $\mathcal{B}_0$. Therefore we use a dynamic programming approach as follows.
With Lemma 2.1 we conclude that the number of different rounded bin sizes in $\mathcal{B}_0$ is at most $O(1/\delta \log(g(1/\delta)))$ and the number q of different large job sizes is at most $O(1/\delta \log(g(1/\delta)/\delta))$. Since $g(1/\delta) = poly(1/\delta)$ both numbers are in $O(1/\delta \log(1/\delta))$. Furthermore, the total number of large jobs packed into $\mathcal{B}_0$ is bounded by $(K/\delta)g(1/\delta)$. A possible choice of large jobs for the first K bins can be described by a vector $v = (v_1, \ldots, v_q)$ where v_j is the number of jobs with processing time r_j packed into $\mathcal{B}_0$. Since $v_j \leq Kg(1/\delta)/\delta$, $g(1/\delta) \in O(poly(1/\delta))$ and $K \leq f(1/\delta) = O(C(\tilde{A}_\delta))$, the total number of possible vectors is at most

$$2^{O(1/\delta \log(g(1/\delta)/\delta) \log(Kg(1/\delta)/\delta))} \leq 2^{O(1/\delta \log(1/\delta) \log(C(\tilde{A}_\delta))}.$$

By a dynamic program we compute the set of all possible vectors and corresponding packings into the K bins with running-time $2^{O(1/\delta \log(1/\delta) \log(C(\tilde{A}_\delta)))} = 2^{O(1/\delta^2 \log^3(1/\delta))}$, which improves to $2^{O(1/\delta \log^2(1/\delta))}$, if $K \in O(1/\delta)$. Simply go over the machines and store the possible vectors for the first i machines for $i = 1, \ldots, K$. Notice that all huge jobs with processing time larger than or equal to $c_{\min}(\mathcal{B}_0)$ have to be placed into $\mathcal{B}_0$. Consequently, a vector v is feasible, if v corresponds to a packing into the K bins and if all huge jobs are placed into $\mathcal{B}_0$. Let S_0 be the free space $\sum_{i=1}^{K} \bar{c}_i - \sum_{j=1}^{q} r_j v_j$ in $\mathcal{B}_0$.

Algorithm 2 Algorithm for case 2

1: Obtain 2 - approximation using the LPT algorithm and compute a value $T \in [LPT(\mathcal{I})/2, LPT(\mathcal{I})]$.
2: Round the processing times and bin capacities and get rounded instance $\bar{\mathcal{I}}$.
3: Preprocess assignment of large jobs ($\bar{p}_j \geq \delta$) for $\mathcal{B}_0$ via dynamic programming and obtain feasible assignment vector v.
4: **if** the dynamic program does not find a feasible solution for $\mathcal{B}_0$ **then**
5: increase T and go to step 2.
6: **end if**
7: Allocate large jobs fractionally into $\mathcal{B}_1$ via an LP.
8: **if** the LP does not have a feasible solution **then**
9: go back to step 3 (and compute a different vector v).
10: **end if**
11: Round solution of the LP with new rounding technique using a subroutine for bin packing with different bin sizes.
12: Schedule the small jobs ($\bar{p}_j < \delta$) behind the large ones slightly increasing the bins sizes.

Allocating large jobs into $\mathcal{B}_1$. Notice that tiny jobs with processing times $\bar{p}_j < \delta c_{\min}(\mathcal{B}_0)$ can be neglected. These jobs can be placed greedily into slightly enlarged bins in the last phase of the algorithm. Now we choose and pack the remaining jobs for S_0 and $\mathcal{B}_1$ via a linear program relaxation similar as in the case of identical machines. Here the gap $g(1/\delta)$ will be helpful to move some jobs later into $\mathcal{B}_0$.
We divide the set of all bins in $\mathcal{B}_1$ into N groups B_ℓ with m_ℓ bins of the same size $\bar{c}(\ell)$. For simplicity we use $B_0 = \mathcal{B}_0$. In the LP below we use a variable $x_i^{(\ell)}$ to indicate the fractional length of a multiset $C_i^{(\ell)}$ of large processing times $\bar{p}_j \in [\delta\bar{c}(\ell), \bar{c}(\ell)]$ packed into bins of size $\bar{c}(\ell)$. Let $a(j, i^{(\ell)})$ be the number of the occurrences of $\bar{p}_j$ in $C_i^{(\ell)}$ and let $size(C_i^{(\ell)}) = \sum_j a(j, i^{(\ell)})\bar{p}_j$. Furthermore, let n_j be the number of jobs with processing time $\bar{p}_j = \delta(1+\delta)^j$ for $j = 0, \ldots, R$ (where $\delta(1+\delta)^R \leq c_{\min}(\mathcal{B}_0)$ is the largest non-huge processing time). According to Lemma 2.1 $R = O(1/\delta \log(c_{\min}(\mathcal{B}_0)/\delta c_{\min}(\mathcal{B}_0)) = O(1/\delta \log(1/\delta))$.

Finally, we use a variable $y_{j,\ell}$ to indicate the fractional number of jobs of size $\delta \leq \bar{p}_j < \delta\bar{c}(\ell)$ packed as a small job in B_ℓ. For a job size with $\bar{p}_j \geq \delta\bar{c}(\ell)$ (that is large with respect to B_ℓ) we set $y_{j,\ell} = 0$. For jobs with $\bar{p}_j \in (c_{\max}(\mathcal{B}_1), c_{\min}(\mathcal{B}_0))$ we have $y_{j,0} = n_j$. Furthermore, we may suppose that $y_{j,0} = 0$ for jobs with processing time at least $\delta c_{\min}(\mathcal{B}_0)$. This means that the large jobs for $\mathcal{B}_0$ are placed into $\mathcal{B}_0$ only in the preprocessing step. This assumption is useful for the rounding of the LP.

$$
\begin{array}{llll}
\text{LP}(\mathcal{B}_1): & & & \\
\sum_i x_i^{(\ell)} & \leq & m_\ell & \text{for } \ell = 1, \ldots, N \\
\sum_{\ell,i} a(j, i^{(\ell)}) x_i^{(\ell)} + \sum_\ell y_{j,\ell} & = & n_j & \text{for } j = 0, \ldots, R \\
\sum_i size(C_i^{(\ell)}) x_i^{(\ell)} + \sum_j y_{j,\ell}\delta(1+\delta)^j & \leq & m_\ell \bar{c}(\ell) & \text{for } \ell = 1, \ldots, N \\
\sum_{j=1}^R y_{j,0}\delta(1+\delta)^j & \leq & S_0 &
\end{array}
$$

$$
\begin{array}{lll}
x_i^{(\ell)} & \geq 0 & \text{for } \ell = 1, \ldots, L \text{ and } i = 1, \ldots, h_\ell \\
y_{j,\ell} & \geq 0 & \text{for } j = 0, \ldots, R \text{ and } \ell = 0, \ldots, N
\end{array}
$$

As in [12] the coefficients in the last $N+1$ constraints of the LP can be rounded and each inequality can be transformed so that the converted coefficients are bounded by $O(n/\delta)$. The facet complexity of the polyhedron corresponding to the resulting dual linear program is bounded by $(O(R\log(n) + n\log(n/\delta))) = O(n\log(n/\delta))$. The encoding length of the vector of the objective function of the dual is bounded by $poly(n, 1/\delta)$. Thus, according to [7] we achieve a basic optimum solution of LP($\mathcal{B}_1$) in time $poly(n, 1/\delta)$.

New rounding technique. We describe here a new approach to round the solution of the LP using a subroutine for bin packing with different bin sizes. Actually, we could also use another rounding method as in [12]. But it requires the gap $g(1/\delta)$ to be much larger: In order to bound the additional $O(1/\delta^2 \log(1/\delta))$ bins of size $c_{\max}(\mathcal{B}_1)$ by the term $\delta c_{\max}(\mathcal{B}_0)$, the gap $g(1/\delta)$ should be at least $\Omega(1/\delta^3 \log(1/\delta))$. So recall the conditions of Case 2, that means that there exists an index $K+1 \leq f(1/\delta) = O(C(\tilde{A}_\delta))$ with $\bar{c}_1/\bar{c}_i < g(1/\delta)$ for $1 \leq i \leq K$ with $g(1/\delta) \geq 1/\delta \log^2(1/\delta)$ and $\bar{c}_1/\bar{c}_i \geq g(1/\delta)$ for $K+1 \leq i \leq m$.

In our new approach we subdivide $\mathcal{B}_1$ into groups of bins $D_1, \ldots, D_H$ with similar bin sizes. These groups are not necessary equal to the groups $B_1, \ldots, B_N$ we considered to set up the above LP-relaxation. Then we use the solution of the LP relaxation above to pack the jobs or items via a bin packing algorithm. For each group D_k the bin packing algorithm packs the selected items into the group D_k of bins with different bin sizes plus few additional bins of maximum capacity $c_{\max}(D_k)$. Based on the subdivision the number of large item sizes can be bounded by $d = O(1/\delta \log(1/\delta))$ for each group D_k. Using a recent result [13], we are able to pack the selected items into D_k plus $O(\log^2(d)) = O(\log^2(1/\delta))$ bins of capacity $c_{\max}(D_k)$. The overall goal is to obtain a packing of almost all jobs into $\mathcal{B}_1$ plus at most $O(\log^2(1/\delta))$ bins of capacity $c_{\max}(\mathcal{B}_1)$. Finally we use the gap $g(1/\delta)$ to move the jobs which lie in the additional bins onto the faster machines with slightly increased makespan. In the following we explain in detail how this rounding works.

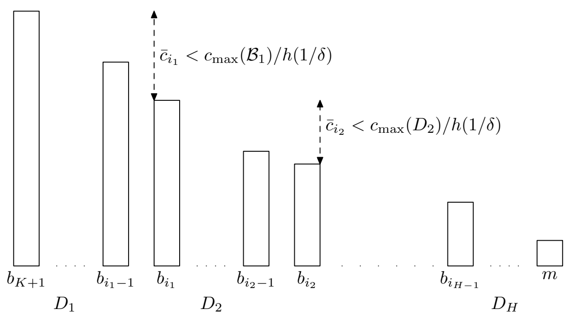

Figure 3: Grouping $\mathcal{B}_1$ into D_1 to D_H

Suppose that $\mathcal{B}_1$ has a bin b_{i_1} with $\bar{c}_{i_1} < c_{\max}(\mathcal{B}_1)/h(1/\delta)$, where $h : \mathbb{R}^+ \to \mathbb{R}^+$ is a function with $poly(1/\delta) \geq h(1/\delta) \geq 1/\delta$. W.l.o.g. let $i_1 \in \{K+1, \ldots, m\}$ be minimal with that property. In this case we build $D_1 = \{b_{K+1}, \ldots, b_{i_1-1}\}$ and construct the other groups $D_2, \ldots, D_H$ iteratively in the same way. The next group $D_2 = \{b_{i_1}, \ldots, b_{i_2-1}\}$ fulfills the properties $c_{\min}(D_2) = \bar{c}_{i_2-1} \geq c_{\max}(D_2)/h(1/\delta)$ and $\bar{c}_{i_2} < c_{\max}(D_2)/h(1/\delta)$, see Figure 3. If all bins have capacity larger than or equal $c_{\max}(\mathcal{B}_1)/h(1/\delta)$, we have only one group $D_1 = \{b_{K+1}, \dots, b_m\}$. Furthermore, if bin $b_{k_j} \in D_k$ has capacity $\bar{c}_\ell$ for $\ell \in \{1, \dots, N\}$ we have $B_\ell \subset D_k$. With Lemma 2.1 we conclude that the number of different bin sizes in each group D_k is at most $O(1/\delta \log(h(1/\delta)))$ and the number of large job sizes in D_k is at most $O(1/\delta \log(h(1/\delta)/\delta))$. Since $h(1/\delta) \leq poly(1/\delta)$, both numbers are bounded by $O(1/\delta \log(1/\delta))$.

Consider now a linear program solution $(x_i^{(\ell)}, y_{j,\ell})$ and consider the reduced linear program LP_k with corresponding constraints for the bin group D_k.

$$
\begin{array}{lcll}
\mathrm{LP}_k : & & & \\
\sum_i x_i^{(\ell)} & = & \bar{m}_\ell & \text{for bins in capacity } c(\ell) \text{ in } D_k \\
\sum_{\ell,i} a(j, i^{(\ell)}) x_i^{(\ell)} & \geq & n_j^{(k)} & \text{for each large job size in } D_k \\
x_i^{(\ell)} & \geq & 0 & \text{for } \ell = 1, \dots, L,
\end{array}
$$

The value $\bar{m}_\ell \leq m_\ell$ is the fractional number of bins of size $\bar{c}(\ell)$ in D_k and $n_j^{(k)}$ is the fractional number of large job sizes $\delta(1+\delta)^j$ placed into D_k according to the solution of LP($\mathcal{B}_1$). A job size $\bar{p}_j$ is large in D_k if $\bar{p}_j = \delta(1+\delta)^j \in [\delta\bar{c}(\ell), \bar{c}(\ell)]$ for at least one bin group B_ℓ in D_k with capacity $\bar{c}(\ell)$. If in LP_k we replace the right hand sides $n_j^{(k)}$ by $\lfloor n_j^{(k)} \rfloor$ for each large job size, we have to cover an integral number of jobs per large job size. Thus, the total execution time $\sum_{\bar{p}_j \text{large in } D_k} \delta(1+\delta)^j$ of the non-covered large jobs in D_k can be bounded by $c_{\max}(D_k) \sum_{j=0}^{\infty} (1+\delta)^{-j} = c_{\max}(D_k)(1+\delta)/\delta$ (using the geometric sum over the job sizes). The analysis for the first block D_1 with bound $c_{\max}(D_1)(1+\delta)/\delta$ for the non-covered jobs above can be even further improved:

Lemma 3.4. *If we replace $n_j^{(1)}$ by $\lfloor n_j^{(1)} \rfloor$ in LP_k, the total processing time of the non-covered large jobs in D_1 is bounded by $(1+\delta)\min\{c_{\min}(\mathcal{B}_0), {}^{c_{\max}(D_1)}/_{\delta}\}$.*

Proof. All jobs with processing time $\bar{p}_j \in (c_{\max}(D_2), \delta c_{\min}(\mathcal{B}_0))$ are large or small corresponding to bins in D_1, but small corresponding to $\mathcal{B}_0$. Notice, that jobs with $\bar{p}_j \in [\delta c_{\min}(\mathcal{B}_0), c_{\max}(D_1)]$ are large corresponding to $\mathcal{B}_0$ and chosen via the preprocessing step above. This implies that we may assume $y_{j,0} = 0$ for the corresponding values. Since $c_{\min}(\mathcal{B}_0) > c_{\max}(D_1)$ and $c_{\max}(D_1)/c_{\max}(D_2) > h(1/\delta) \geq 1/\delta$, we have $c_{\max}(D_2) < \delta c_{\max}(D_1) < \delta c_{\min}(\mathcal{B}_0)$. Therefore, a job with $\bar{p}_j \in [\delta c_{\min}(\mathcal{B}_0), c_{\max}(D_1)]$ does not fit into D_2. Moreover, we can assume that the values $n_j^{(1)}$ are integral. Since the values $n_j^{(1)}$ are integral for processing times $\bar{p}_j \in [\delta c_{\min}(\mathcal{B}_0), c_{\max}(D_1)]$, the not covered large jobs have processing times $\leq \delta c_{\min}(\mathcal{B}_0)$. Consequently, the total sum of the execution times of the large jobs not covered by the bin packing subroutine in $\mathcal{D}_1$ is bounded by $\min\{(1+\delta)c_{\min}(\mathcal{B}_0), c_{\max}(D_1)(1+\delta)/\delta\}$ (as above using the geometric sum argument applied to $\delta c_{\min}(\mathcal{B}_0)$ instead of $c_{\max}(\mathcal{D}_k)$). □

Now a (fractional) solution of the modified LP_k can be transformed into an integral solution. That means $\lfloor n_j^{(k)} \rfloor$ jobs of size $\delta(1+\delta)^j$ can be packed into the bins in D_k plus $O(\log^2(d))$ additional bins of size $c_{\max}(D_k)$ [13] (where $d = O(1/\delta \log(1/\delta))$ is the number of different large job sizes). Notice that it is allowed to use m_ℓ bins instead of the fractional number $\bar{m}_\ell$ of bins in each group B_ℓ. This is sufficient, since the overall area $\sum_{\ell:B_\ell\subset D_k} Area(large,\ell)$ of the large jobs packed into D_k plus the extra bins remains the same.

The $(y_{j,\ell})$ variables can be rounded as before [12] using a result of Lenstra et al. [17], but here we have to round up (instead of down) the values N_j, the number of jobs of size $\delta(1+\delta)^{k_j}$ assigned as a small job to the blocks B_ℓ for all job sizes j. Almost all corresponding small jobs can be packed directly into $\mathcal{B}_1$. For each block B_ℓ there is at most one fractional variable $\tilde{y}_{j,\ell}$ and a corresponding small job that does not fit completely in B_ℓ. But this job can be packed into one bin of B_ℓ, if we increase the corresponding bin size slightly.

Lemma 3.5. *The total execution time of the jobs in the additional bins for $D_1,\ldots,D_H$ is at most $(1+\delta)\min\{c_{\min}(\mathcal{B}_0),(1/\delta)c_{\max}(D_1)\}+O(\log^2(1/\delta))c_{\max}(D_1)$.*

Proof. The total execution time of all jobs in the additional bins for D_k is at most

$$[\frac{1+\delta}{\delta}+O(\log^2(1/\delta))]c_{\max}(D_k)\le\frac{C}{\delta}c_{\max}(D_k)$$

where C is a constant. Using $h(1/\delta)c_{\max}(D_{k+1})\le c_{\max}(D_k)$ this implies the following bound for the extra execution time in $D_2,\ldots,D_H$

$$\begin{aligned}\textstyle\sum_{k\ge2}\frac{C}{\delta}c_{\max}(D_k)\le\frac{C}{\delta}\sum_{k\ge0}\frac{c_{\max}(D_2)}{h(1/\delta)^k}&\stackrel{h(1/\delta)\ge1/\delta}{\le}\textstyle\frac{C}{\delta}c_{\max}(D_2)\sum_{k\ge0}\delta^k\\&\stackrel{\delta\le1/2}{\le}\frac{2C}{\delta}c_{\max}(D_2)\le O(1)c_{\max}(D_1).\end{aligned}$$

With Lemma 3.4 the total extra execution time among the bin groups $D_1,\ldots,D_H$ is at most $(1+\delta)\min\{c_{\min}(\mathcal{B}_0),(1/\delta)c_{\max}(D_1)\}+O(\log^2(1/\delta))c_{\max}(D_1)+O(1)c_{\max}(D_1)$. □

Notice that all jobs involved in the first term are small corresponding to $\mathcal{B}_0$ (i.e. $\bar{p}_j\le\delta c_{\min}(\mathcal{B}_0)$). The other jobs that contribute to $O(\log^2(1/\delta))c_{\max}(D_1)$ could be large corresponding to $\mathcal{B}_0$, but have processing time $\bar{p}_j\le c_{\max}(D_1)=c_{\max}(\mathcal{B}_1)$. We can now distribute the jobs corresponding to the additional term among the bins of $\mathcal{B}_0$. We proceed by case distinction.

Case 2.1: $K\ge\lceil(1+\delta)/\delta\rceil$.
In this case the additional term $(1+\delta)c_{\min}(\mathcal{B}_0)$ with small jobs of size at most $\delta c_{\min}(\mathcal{B}_0)$ can be distributed onto the first $\lceil(1+\delta)/\delta\rceil$ bins with additional load $\le 2\delta c_{\min}(\mathcal{B}_0)$. Here simply use a greedy algorithm that distributes a load of at least $\delta c_{\min}(\mathcal{B}_0)$ and at most $2\delta c_{\min}(\mathcal{B}_0)$ (if possible) on the first bins. The last bin gets load at most $\delta c_{\min}(\mathcal{B}_0)$. The number of bins needed is at most $\lceil(1+\delta)/\delta\rceil\le K$. This implies that the first K bins get load at most $\bar{c}_i+2\delta c_{\min}(\mathcal{B}_0)\le(1+2\delta)\bar{c}_i$. This implies the following result

Lemma 3.6. *Let b_{K+1} be the first bin such that $\bar{c}_1/\bar{c}_{K+1}\ge g(1/\delta)$ and $K\ge\lceil(1+\delta)/\delta\rceil$, then we can compute an approximate schedule of length $(1+O(\delta))OPT(I)$ in time $2^{O(1/\delta\log(1/\delta)\log(C(\tilde{A}_\delta)/\delta))}poly(1/\delta,n)$.*

Proof. Since $g(1/\delta) \geq 1/\delta \log^2(1/\delta)$ and $\frac{c_1}{c_{K+1}} \geq g(1/\delta)$ we can schedule the remaining jobs of length $\bar{p}_j \leq c_{\max}(\mathcal{B}_1)$ with total execution time $O(\log^2(1/\delta))c_{\max}(D_1)$ into $\mathcal{B}_0$ in the first largest bin: We have $\log^2(1/\delta)c_{\max}(\mathcal{B}_1) \leq \frac{\log^2(1/\delta)}{g(1/\delta)}c_{\max}(\mathcal{B}_0) \leq O(\delta)c_{\max}(\mathcal{B}_0)$. $\square$

Case 2.2: $c_{max}(\mathcal{B}_0)/c_{max}(\mathcal{B}_1) \geq 1/\delta^2$.
In this case the additional processing time is bounded by

$$((1+\delta)/\delta)c_{\max}(D_1) + O(\log^2(1/\delta))c_{\max}(D_1) \leq O(1/\delta)c_{\max}(D_1) \leq O(\delta)c_{\max}(\mathcal{B}_0).$$

This amount can be packed onto the largest bin by increasing its bin size to $(1 + O(\delta))c_{\max}(\mathcal{B}_0)$. We summarize

Lemma 3.7. *Let $g(1/\delta) \geq 1/\delta^2$, then we can compute an approximate schedule of length $(1+O(\delta))OPT(I)$ in time $2^{O(1/\delta \log(1/\delta)\log(C(\tilde{A}_\delta)/\delta))}poly(1/\delta, n)$.*

Case 2.3: $K < \lceil (1+\delta)/\delta \rceil$ and $c_{max}(\mathcal{B}_0)/c_{max}(\mathcal{B}_1) < 1/\delta^2$.
Notice that we can also use our general assumption on $g(1/\delta)$. This implies that $\bar{c}_1/\bar{c}_{K+1} \geq 1/\delta \log^2(1/\delta)$. Here let us consider jobs with processing time $\bar{p}_j$ in the interval $I = [\delta c_{\max}(D_1), \min\{\delta c_{\min}(\mathcal{B}_0), c_{\max}(D_1)\}]$ and denote with $P_{cri} = \{\bar{p}_j \in I\}$ the processing times of those critical jobs. Since $\max(P_{cri})/\min(P_{cri}) \leq 1/\delta$, there are at most $O(1/\delta \log(1/\delta))$ different rounded processing times for the critical jobs. Furthermore, the total number of critical jobs that can be packed into $\mathcal{B}_0$ is at most $Kc_{\max}(\mathcal{B}_0)/(\delta c_{\max}(D_1)) \leq O(1/\delta^2)c_{\max}(\mathcal{B}_0)/c_{\max}(D_1) \leq O(1/\delta^4)$ under the second assumption above.

The critical jobs can be chosen for $\mathcal{B}_0$ and packed into the bins in $\mathcal{B}_0$ during our preprocessing step. For $g(1/\delta) = O(1/\delta \log^2(1/\delta))$, the large and critical jobs can be selected and pre-packed into $\mathcal{B}_0$ in time $(1/\delta^4)^{O(1/\delta \log(1/\delta))}$
$\leq 2^{O(1/\delta \log^2(1/\delta))}$. For each corresponding feasible vector v we set up a linear program, round the solution and place the corresponding large jobs into the bins plus few additional bins of size $O(\log^2(1/\delta))c_{max}(D_1) \leq O(\log^2(1/\delta))c_{\max}\mathcal{B}_0/g(1/\delta) \leq O(\delta)c_{max}(\mathcal{B}_0)$. In this case, for all critical jobs with $\bar{p}_j \in P_{cri}$ we may suppose that $y_{j,0} = 0$ (no further such jobs are packed into $\mathcal{B}_0$ via the LP). This implies that the corresponding values $n_j^{(1)}$ are integral. Here again we use the fact that a large job in D_1 with $\bar{p}_j \geq \delta c_{max}(D_1) > c_{max}(D_2)$ does not fit into bin group D_2. In this case the term for the non-covered jobs in D_1 can be bounded by $(1+\delta)c_{max}(D_1) \leq \delta c_{max}(\mathcal{B}_0)$. Furthermore, note that for $K = O(1/\delta)$, the running time of our algorithm is bounded by $2^{O(1/\delta \log^2(1/\delta))}poly(1/\delta, n)$ (compare to Section 3.2). This gives

Lemma 3.8. *Let b_{K+1} be the first bin such that $\bar{c}_1/\bar{c}_{K+1} \geq \Omega(1/\delta \log^2(1/\delta))$, $K < \lceil (1+\delta)/\delta \rceil$ and $\bar{c}_1/\bar{c}_{K+1} < 1/\delta^2$, then we can compute an approximate schedule of length $(1+O(\delta))OPT(I)$ in time $2^{O(1/\delta \log^2(1/\delta))}poly(1/\delta, n)$.*

The analysis of all three subcase implies the following result.

Theorem 3.9. *In each subcase of case 2 the algorithm produces a schedule of length $(1+O(\delta))OPT(\mathcal{I})$ in time $2^{O(1/\delta \log(1/\delta)\log(C(\tilde{A}_\delta))}poly(1/\delta, n)$.*

3.3 Algorithm for Case 3

In this case we have two or three bin groups depending on the shape of bin sizes as depicted in Figure 4. Let $\mathcal{B}_0 = \{b_1, \ldots, b_K\}$ be the set of the largest bins. Then, we define $\mathcal{B}_1 = \{b_i | i > K, \bar{c}_i \geq \delta c_{min}(\mathcal{B}_0)\}$ and $\mathcal{B}_2 = \mathcal{B} \setminus (\mathcal{B}_0 \cup \mathcal{B}_1)$ as the remaining bins. If $\mathcal{B}_1 \neq \emptyset$ we distinguish large, medium and small jobs. A job is called *large* if $\bar{p}_j > \delta c_{min}(\mathcal{B}_0)$ and *medium* if $\bar{p}_j \in (\delta c_{min}(\mathcal{B}_1), \delta c_{min}(\mathcal{B}_0)]$; other jobs are called *small.* Note that for a medium job we have $\bar{p}_j \leq \delta c_{\min}(\mathcal{B}_0) \leq c_{\max}(\mathcal{B}_1)$ by construction. If $\mathcal{B}_1 = \emptyset$ we do not have medium jobs. In this case we have for all $i > K$ that $\bar{c}_i \leq \delta c_{\min}(B_0)$. Thus, we have an additional gap between $\bar{c}_K$ and $\bar{c}_{K+1}$, i.e. $\frac{\bar{c}_K}{\bar{c}_{K+1}} = \frac{c_{\min}(\mathcal{B}_0)}{c_{\max}(\mathcal{B}_1)} > \frac{c_{\min}(\mathcal{B}_0)}{\delta c_{\min}(\mathcal{B}_0)} = \frac{1}{\delta}$.

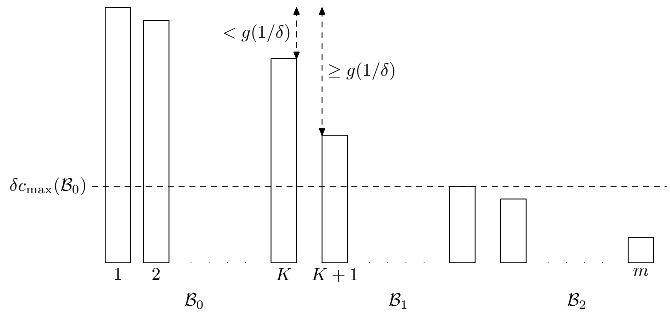

Figure 4: Shape of bins for case 3

Lemma 2.1 implies that the number of different rounded bin sizes and large and medium job sizes corresponding to $\mathcal{B}_0 \cup \mathcal{B}_1$ is bounded by $O(1/\delta \log(g(1/\delta)/\delta))$ and $O(1/\delta \log(g(1/\delta)/\delta^2))$, respectively. Notice that both numbers are at most $O(1/\delta \log(1/\delta))$ since $g(1/\delta) = poly(1/\delta)$.
Now we divide the set $\mathcal{B} = \mathcal{B}_0 \cup \mathcal{B}_1 \cup \mathcal{B}_2$ into N groups B_ℓ with m_ℓ bins with the same rounded bin size $\bar{c}(\ell)$ for $\ell = 1, \ldots, N$ and set up a linear program. Later we consider a reduced LP for the first two bin groups separately. In the LP below we use a variable $x_i^{(\ell)}$ to indicate the fractional length of a multiset $C_i^{(\ell)}$ of large processing times $\bar{p}_j \in [\delta\bar{c}(\ell), \bar{c}(\ell)]$ packed into bins of size $\bar{c}(\ell)$. Let $a(j, i^{(\ell)})$ be the number of the occurrences of $\bar{p}_j$ in $C_i^{(\ell)}$ and let $size(C_i^{(\ell)}) = \sum_j a(j, i^{(\ell)})\bar{p}_j$. Furthermore, let n_j be the number of jobs with processing time $\bar{p}_j = \delta(1+\delta)^j$ for $j = 0, \ldots, R$ (where $\delta(1+\delta)^R$ is the largest jobs size). Finally, we use a variable $y_{j,\ell}$ to indicate the fractional number of jobs of size $\delta \leq \bar{p}_j < \delta\bar{c}(\ell)$ packed as a small job in B_ℓ.

$$\begin{array}{lcll}
\sum_i x_i^{(\ell)} & \leq & m_\ell & \text{for } \ell = 1, \ldots, N \\
\sum_{\ell,i} a(j, i^{(\ell)}) x_i^{(\ell)} + \sum_\ell y_{j,\ell} & = & n_j & \text{for } j = 0, \ldots, R \\
\sum_i size(C_i^{(\ell)}) x_i^{(\ell)} + \sum_j y_{j,\ell}\delta(1+\delta)^j & \leq & m_\ell \bar{c}(\ell) & \text{for } \ell = 1, \ldots, N
\end{array}$$

$$\begin{array}{lll}
x_i^{(\ell)} & \geq 0 & \text{for } \ell = 1, \ldots, N \text{ and } i = 1, \ldots, h_\ell \\
y_{j,\ell} & \geq 0 & \text{for } j = 0, \ldots, R \text{ and } \ell = 1, \ldots, N
\end{array}$$

We suppose that all jobs fit into the bins, i.e. $\delta(1+\delta)^R \leq c_{max}(\mathcal{B}_0)$; otherwise there is no schedule with the corresponding makespan in the binary search. Suppose that $\mathcal{B}_0$ consists of L bin groups B_ℓ and $\mathcal{B}_1$ consists of P bin groups, see also Fig. 5.

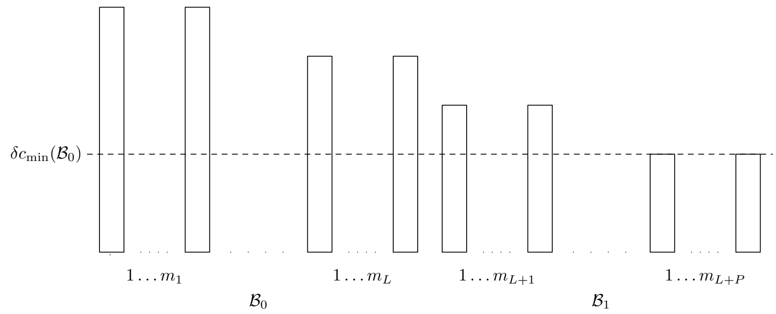

Figure 5: Groups of similar capacities in $\mathcal{B}_0 \cup \mathcal{B}_1$

Allocating large jobs into $\mathcal{B}_0$ and $\mathcal{B}_1$. Suppose that the entire LP and the corresponding ILP have a solution. Consider now the corresponding $(x_i^{(\ell)})$ variables and constraints for the first $L+P$ bin groups. Let $\delta(1+\delta)^{R_m}$ be the smallest medium job size and let $\delta(1+\delta)^{R_\ell}$ be the smallest large job size.

$$\begin{array}{lll} \sum_i x_i^{(\ell)} & \leq & \bar{m}_\ell \leq m_\ell \quad \text{for} \quad \ell = 1, \ldots, L+P \\ \sum_{\ell,i} a(j, i^{(\ell)}) x_i^{(\ell)} & \geq & \bar{n}_j \qquad\quad\;\; \text{for} \quad j = R_m, \ldots, R \end{array}$$

$$x_i^{(\ell)} \geq 0 \qquad \text{for} \quad \ell = 1, \ldots, L+P \quad \text{and} \quad i = 1, \ldots, h_\ell$$

For a large job size we have $\bar{p}_j > \delta c_{min}(\mathcal{B}_0) > c_{\max}(\mathcal{B}_2)$. Hence, the large jobs have to be scheduled in $\mathcal{B}_0 \cup \mathcal{B}_1$. Consequently, we describe them by configuration variables only in the original LP and so the number $\bar{n}_j$ for large jobs covered by the LP above is integral and satisfies $\bar{n}_j = n_j$.
For medium job sizes, there are $y_{j,\ell}$ variables in the initial LP and we have in general fractional variables $\bar{n}_j \leq n_j$. Note that a configuration $C_i^{(\ell)}$ in $\mathcal{B}_0$ contains only large job sizes by construction and a configuration $C_i^{(\ell)}$ in $\mathcal{B}_1$ may contain both, large and medium job sizes.

Let $\bar{C}_k^{(\ell)}$ be a configuration with only large job sizes in bin group B_ℓ in $\mathcal{B}_1$ and let $z_k^{(\ell)}$ be a variable that indicates the total length of $\bar{C}_k^{(\ell)}$. For the rest of the paper we call $\bar{C}_k^{(\ell)}$ a big configuration. Then, the original configurations with both, medium and large job sizes, can be partitioned into groups with the same arrangement of large jobs according to configuration $\bar{C}_k^{(\ell)}$ (containing only large jobs). Let $Index(k, \ell)$ be the set of all indices i such that $C_i^{(\ell)}$ coincides with $\bar{C}_k^{(\ell)}$ for the large job sizes. Then, $z_k^{(\ell)} = \sum_{i \in Index(k,\ell)} x_i^{(\ell)}$ and the following modified LP for the large job sizes has a feasible solution.

$$\begin{array}{lll} \text{LP}_{large} & & \\ \sum_i x_i^{(\ell)} & \leq & m_\ell \quad \text{for} \quad \ell = 1, \ldots, L \\ \sum_k z_k^{(\ell)} & \leq & m_\ell \quad \text{for} \quad \ell = L+1, \ldots, L+P \\ \sum_{\ell,i} a(j, i^{(\ell)}) x_i^{(\ell)} + \sum_{\ell,k} a(j, k^{(\ell)}) z_k^{(\ell)} & \geq & n_j \quad \text{for} \quad j = R_\ell, \ldots, R \end{array}$$

$$x_i^{(\ell)} \geq 0 \qquad \text{for} \quad \ell = 1, \ldots, L+P \quad \text{and} \quad i = 1, \ldots, h_\ell$$

Since all large job sizes have to be placed into the first $L+P$ bin groups and using the assumption that the entire ILP has a solution, the modified ILP for the large job sizes has a feasible solution, too. Using the Theorem by Cook et al. [3], there is an ILP

solution $(\hat{x}_i^{(\ell)}, \hat{z}_k^{(\ell)})$ with distances $\|\hat{x}_i^{(\ell)} - x_i^{(\ell)}\|_\infty$ and $\|\hat{z}_k^{(\ell)} - z_k^{(\ell)}\|_\infty$ bounded by $N\Delta \leq 2^{O(1/\delta \log^2(g(1/\delta)/\delta+1))}$. Notice that the column sum of a column of the constraint matrix $\sum_j a(j, i^{(\ell)}) + 1$ corresponding to a configuration is at most $c_{max}(\mathcal{B}_0)/(\delta c_{min}(\mathcal{B}_0)) + 1 \leq g(1/\delta)/\delta + 1$. Since $g(1/\delta) = poly(1/\delta)$, the distances above are at most
$C(\tilde{A}_\delta) = 2^{O(1/\delta \log^2(1/\delta))}$. If $x_i^{(\ell)}$ or $z_k^{(\ell)}$ is larger than $C(\tilde{A}_\delta)$, then we know that there is an integer solution with $\hat{x}_i^{(\ell)} \geq \lceil x_i^{(\ell)} - C(\tilde{A}_\delta)\rceil$ or $\hat{z}_k^{(\ell)} \geq \lceil z_k^{(\ell)} - C(\tilde{A}_\delta)\rceil$. Then we can reduce our instance $\bar{I}$ to a reduced instance I_{red} with $\tilde{n}$ large jobs and $\tilde{m}_\ell \leq m_\ell$ bins per block B_ℓ as described in the introduction.

The values of the coefficients of the constraint matrix are bounded by the number of large jobs per configuration which is at most $c_{max}(\mathcal{B}_0)/(\delta c_{min}(\mathcal{B}_0)) \leq g(1/\delta)/\delta$. The number of strict positive variables of a basic solution of the modified LP is at most $O(1/\delta \log(g(1/\delta)/\delta))$. Since each reduced variable has value at most $C(\tilde{A}_\delta)$, $\tilde{n}$ can be bounded by $O(1/\delta^2 g(1/\delta) \log(g(1/\delta)/\delta)) C(\tilde{A}_\delta)$. Since $g(1/\delta) = poly(1/\delta)$, the number of remaining large jobs is at most
$\tilde{n} \leq poly(1/\delta) C(\tilde{A}_\delta) 2^{O(\log(1/\delta))} C(\tilde{A}_\delta)$. Moreover, this implies that we need at most $\tilde{M} \leq \tilde{n} \leq 2^{O(1/\delta \log^2(1/\delta))}$ machines for the large jobs in I_{red}. Since the modified ILP for the large sizes has a feasible solution, we can find a solution for I_{red} by dynamic programming. Simply go over the machines in $\mathcal{B}_0 \cup \mathcal{B}_1$ and place the $\tilde{n}$ jobs onto the machines. This can be done by computing feasible vectors $(x_{R_\ell}, \ldots, x_R)$ that correspond to a packing of x_i large jobs of size $\bar{p}_i$ into the first k bins for $k = 1, \ldots, \sum_{\ell=1}^{L+P} \tilde{m}_\ell$. In this way we can find a feasible packing in time $\tilde{n}^{O(1/\delta \log(g(1/\delta)/\delta))} \leq 2^{O(1/\delta \log(1/\delta) \log \tilde{n})} \leq 2^{O(1/\delta \log(1/\delta) \log(C(\tilde{A}_\delta)))}$.

Algorithm 3 Algorithm for case 3

1: Obtain 2 - approximation using the LPT algorithm.
2: Compute a value $T \in [LS(I)/2, LS(I)]$.
3: Round the processing times of the jobs and distinguish small, medium and large jobs
4: Allocate a subset of the large jobs into $\mathcal{B}_0$ and $\mathcal{B}_1$ using a linear program and with Theorem by Cook et al. [3] reduce the instance.
5: **if** the linear program does not have a feasible solution **then**
6:     increase T and go to step 2.
7: **end if**
8: Via dynamic programming obtain an assignment of the remaining large jobs into $\mathcal{B}_0$ and $\mathcal{B}_1$.
9: **if** the dynamic program for I_{red} does not find a feasible solution **then**
10:     increase T and go to step 2.
11: **end if**
12: Allocate medium jobs into $\mathcal{B}_1$ via a bin packing subroutine.
13: Fit the allocated large and medium jobs together with the remaining jobs.
14: Schedule the small jobs behind the large ones. .

Allocating medium jobs. The main difficulty now is to handle the medium jobs. Consider the LP for the medium and large jobs corresponding to bin group $\mathcal{B}_1$. Take out for a moment the large jobs $I_{large,dp}$ placed by the dynamic program into $\mathcal{B}_1$. Notice that these large jobs have occupied a subset $M_{large,dp}$ of only $\tilde{M} \leq \tilde{n}$ machines in $\mathcal{B}_1$. Furthermore, notice that there are still large jobs preassigned via the big configurations $\bar{C}_k^{(\ell)}$ of length $\lceil z_k^{(\ell)} - C(\tilde{A}_\delta)\rceil = \lceil \sum_{i \in Index(k,\ell)} x_i^{(\ell)} - C(\tilde{A}_\delta)\rceil$ in $\mathcal{B}_1$. Since we have a feasible LP solution for all jobs, the residual configurations $C_i^{(\ell)}$ (restricted to medium job sizes) with fractional lengths $x_i^{(\ell)}$ fit into the gaps either besides their corresponding big configurations of lengths $\lceil z_k^{(\ell)} - C(\tilde{A}_\delta)\rceil$ or after them. The placement of medium jobs can be seen as a fractional bin packing problem with different bin sizes. We round the $x_i^{(\ell)}$ variables for $\mathcal{B}_1$ and use our new rounding technique presented in Section 3.2. Since medium jobs are small corresponding to $\mathcal{B}_0$ and since $K \geq f(1/\delta) \geq \lceil (1+\delta+\log^2(1/\delta))/\delta \rceil$ we can distribute medium jobs corresponding to the additional term among the first K bins. Here we use a greedy algorithm that allocates a load of at least $\delta c_{\min}(\mathcal{B}_0)$ and at most $2\delta c_{\min}(\mathcal{B}_0)$ on the first bins. This increases the makespan by at most $2\delta c_{\min}(\mathcal{B}_0)$. It is also possible that the total area of large jobs pre-assigned via the LP to $\mathcal{B}_0$ is smaller than the total area of large jobs placed via the pre-assignment with configuration lengths $\lceil x_i^{(\ell)} - C(\tilde{A}_\delta)\rceil$ and the dynamic program into $\mathcal{B}_0$. This implies that this additional occupied area in $\mathcal{B}_0$ can not be used for medium and small jobs. Then some medium jobs cannot be placed correctly onto the machines. We show in the lemma below how to place these jobs into $\mathcal{B}_0$. Furthermore, some small jobs have to be placed into $\mathcal{B}_1$. But this is easier and possible, since these jobs are small corresponding to the bins in $\mathcal{B}_1$ and the total area of large, medium and small jobs corresponding to the variable values $x_i^{(\ell)}$ and $y_{j,\ell}$ for $\ell = 1, \ldots, K+L$ fits into $\mathcal{B}_0 \cup \mathcal{B}_1$.

Lemma 3.10. *The medium jobs, that do not fit into $\mathcal{B}_0$ because of additional large jobs placed by the dynamic program into $\mathcal{B}_0$, can be distributed among the machines in $\mathcal{B}_0$, so that the makespan is bounded by $(1 + O(\delta))OPT(\mathcal{I})$.*

Proof. Each additional large job placed into $\mathcal{B}_0$ has size at most $c_{max}(\mathcal{B}_1)$ and the total area of these jobs can be bounded by $\tilde{n}c_{max}(\mathcal{B}_1)$. A better bound can be obtained as follows. The number of strict positive variables $z_k^{(\ell)}$ in $\mathcal{B}_1$ is at most $O(1/\delta \log(g(1/\delta)/\delta))$. For each such variable a strip with total area of at most $C(A)size(\bar{C}_k^{(\ell)}) \leq C(A)\bar{c}(\ell)$ with large jobs could be moved via the dynamic program to $\mathcal{B}_0$. This gives a total area of at most $O(1/\delta \log(g(1/\delta)/\delta))C(A)c_{max}(\mathcal{B}_1)$ that can not be used in the worst case for medium jobs. The number of medium jobs that fit into such area and can not be packed into $\mathcal{B}_0$ is bounded by $O(1/\delta \log(g(1/\delta)/\delta))C(A)c_{max}(\mathcal{B}_1)/(\delta c_{min}(\mathcal{B}_1) \leq O(1/\delta^3 \log(g(1/\delta)/\delta))C(A)$. These jobs can be placed separately onto machines in B_0, since the number of machines is large enough. □

Repacking process. Packing the allocated large and medium jobs together with the remaining jobs into the bins requires an extensive repacking process described in the following steps.

Step 1: Remove the set A_{medium} of medium jobs placed onto machines belonging to $M_{large,dp} \subset \mathcal{B}_1$. Reinsert the large jobs from $I_{large,dp}$ onto these machines and place fractionally a subset of A_{medium} into the remaining gaps.

Lemma 3.11. *The schedule produced in Step 1 can be made integral and has makespan at most $(1 + O(\delta))OPT(\mathcal{I})$.*

Proof. If the total area of the large jobs $I_{large,dp}$ placed via the dynamic program is at most the total area occupied by the reduced variables $\tilde{z}_k^{(\ell)} = z_k^{(\ell)} - \lceil z_k^{(\ell)} - C(A) \rceil$, then all medium jobs fit fractionally into the gaps in $\mathcal{B}_1$. Let $A_{medium,fr}$ be the set of fractional medium jobs placed into $\mathcal{B}_1$. Notice that the cardinality $|A_{medium,fr}|$ is at most $\tilde{M} \leq \tilde{n}$. Since $f(1/\delta)\Theta(\tilde{n})$ these jobs can be placed (separately) onto machines in $\mathcal{B}_0$, increasing the makespan by at most $\delta c_{\min}(\mathcal{B}_0) \leq \delta OPT$. If the total area of large jobs is larger than the LP bound, we have to place the remaining set $A'_{medium} \subset A_{medium}$ of medium jobs into $\mathcal{B}_0$. Since we have $A_{medium} \leq M_{large,dp} c_{\max}(\mathcal{B}_1)$, we conclude $|A'_{medium}| \leq \frac{M_{large,dp} c_{\max}(\mathcal{B}_1)}{\delta c_{\min}(\mathcal{B}_1)} \leq \frac{M_{large,dp} c_{\max}(\mathcal{B}_0)}{\delta^2 g(1/\delta) c_{\max}(\mathcal{B}_0)} \leq O(1/\delta^4 \log(1/\delta)) C(A) \leq 2^{1/\delta \log^2(1/\delta)}$. Since $f(1/\delta) \in 2^{\Omega(1/\delta \log^2(1/\delta))}$, the jobs can be distributed among the bins of $\mathcal{B}_0$. □

Step 2: Round the $(x_i^{(\ell)})$ variables corresponding to $\mathcal{B}_2$ and place the jobs via a bin packing subroutine into $\mathcal{B}_2$ plus some additional bins of size $c_{max}(\mathcal{B}_2) \leq \delta c_{\max}(\mathcal{B}_0)$. This can be done via our new rounding technique similar as the medium jobs are placed in $\mathcal{B}_1$. The additional bins can be distributed among the first K bins.
Step 3: As in [12] we round the $(y_{j,\ell})$ variables over the bin groups B_ℓ using a result of Lenstra et al. [17] and place the corresponding jobs greedily onto the machines. Thereby we have to place in addition one fractional job per bin group on one machine. Since the jobs corresponding to $y_{j,\ell}$ are small in $\mathcal{B}_\ell$ we only have to increase the bin sizes slightly.

Thus, we achieve our main theorem

Theorem 3.12. *There is an EPTAS for scheduling jobs on uniform machines with running-time*

$$2^{O(1/\varepsilon \log(1/\varepsilon) \log(C(\tilde{A}_\delta)))} + poly(n) = 2^{O(1/\varepsilon^2 \log^3(1/\varepsilon))} + poly(n).$$

If $C(\tilde{A}_\delta) = poly(1/\varepsilon)$, the running-time improves to $2^{O(1/\varepsilon \log^2(1/\varepsilon))} + poly(n)$.

References

[1] N. Alon, Y. Azar, G.J. Woeginger, and T. Yadid: Approximation schemes for scheduling on parallel machines, *Journal on Scheduling* 1, 1998, 55-66.

[2] E.G. Coffman, M.R. Garey, and D.S. Johnson: An application of bin packing to multiprocessor scheduling, *SIAM Journal on Computing*, 7 (1978), 1-17.

[3] W. Cook, A.M.H. Gerards, A. Schrijver and É. Tardos: Sensitivity theorems in integer linear programming, *Mathematical Programming* 34 (1986), 251-264.

[4] F. Eisenbrand and G. Shmonin: Caratheodory bounds for integer cones, *Operations Research Letters*, 34 (2006), 564-568.

[5] T. Gonzales, O.H. Ibarra, and S. Sahni: Bounds for LPT schedules on uniform processors, *SIAM Journal on Computing*, 6 (1977), 155-166.

[6] R.J. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics*, 5 (1979), 287-326.

[7] M. Grötschel, L. Lovasz, and A. Schrijver: Geometric Algorithms and Combinatorial Optimization, Springer-Verlag, 1987.

[8] D.S. Hochbaum: Various notions of approximations: good, better, best, and more, Cha. 9 in: *Approximation Algorithms for NP-Hard Problems* (D.S. Hochbaum, ed.), Prentice Hall, 1997, 346- 398.

[9] D.S. Hochbaum and D.B. Shmoys: Using dual approximation algorithms for scheduling problems: practical and theoretical results, *Journal of the ACM*, 34 (1987), 144-162.

[10] D.S. Hochbaum and D.B. Shmoys: A polynomial approximation scheme for scheduling on uniform processors: using the dual approximation approach, *SIAM Journal on Computing* 17 (1988), 539-551.

[11] R. Horowitz and S. Sahni: Exact and approximate algorithms for scheduling non-identical processors, *Journal of the ACM*, 23 (1976), 317-327.

[12] K. Jansen: An EPTAS for scheduling jobs on uniform processors: using an MILP relaxation with a constant number of integral variables, *SIAM J. Discrete Math.* 24(2): 457-485 (2010).

[13] K. Jansen: A fast approximation scheme for the multiple knapsack problem, *To appear in: International Conference on Current Trends in Theory and Practise of Computer Science, SOFSEM 2012.*

[14] K. Jansen and C. Robenek: Scheduling on uniform processors revisited, Technical Report, University of Kiel.

[15] R. Kannan: Minkowski's convex body theorem and integer programming, *Mathematics of Operations Research* 12 (1987), 415- 440.

[16] H.W. Lenstra: Integer programming with a fixed number of variables, *Mathematics of Operations Research* 8 (1983), 538-548.

[17] J.K. Lenstra, D.B. Shmoys, and E. Tardos: Approximation algorithms for scheduling unrelated parallel machines, *Mathematical Programming*, 24 (1990), 259-272.

[18] J. Leung: Bin packing with restricted piece sizes, *Information Processing Letters*, 31 (1989), 145-149.