INSTITUT FÜR INFORMATIK

Faster EPTAS for Scheduling on Uniform Machines

Klaus Jansen, Björn Schumacher, Roberto Solis-Oba

Bericht Nr. 2501
September 2025
ISSN 2192-6247

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

ZU KIEL

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

Faster EPTAS for Scheduling on Uniform Machines

Klaus Jansen, Björn Schumacher, Roberto Solis-Oba

Bericht Nr. 2501
September 2025
ISSN 2192-6247

e-mail: {kj, bsch}@informatik.uni-kiel.de, solis@csd.uwo.ca

Technical Report

Abstract

We present an efficient polynomial-time approximation scheme (EPTAS) for the problem of scheduling jobs on parallel uniform machines with runtime $2^{\mathcal{O}(\varepsilon^{-1}\log^2\varepsilon^{-1})} + \mathcal{O}(n)$, for any constant $\varepsilon > 0$, that improves all previously known EPTAS for the problem. Our algorithm uses a Mixed Integer Linear Program (MILP) formulation for a relaxed version of the problem. We simplify the MILP by repeatedly removing carefully selected sets of jobs and machines until all integer variables have been removed. Then, we build up the solution through the use of a dynamic programming approach, enumeration, and linear program solving. Notable about our approach is that our algorithm only uses an LP solver, different from previous EPTAS approaches where MILP solvers are necessary.

1 Introduction

Given n jobs with processing times p_j for $j \in [n]$[1] and m machines with speeds s_i for $i \in [m]$, where executing a job J_j on a machine M_i takes time $\frac{p_j}{s_i}$, the problem that we consider is to schedule the jobs on the machines to minimize the makespan or maximum completion time. This classical problem is denoted as $Q||C_{\max}$ in the three-field notation of Graham, Lawler, Lenstra, and Kan [8]. Formally, the problem is to find an assignment $\sigma : [n] \to [m]$ of jobs to machines that minimizes $\max_{i\in[m]} \sum_{j\in\sigma^{-1}(i)} \frac{p_j}{s_i}$. $Q||C_{\max}$ is NP-hard in the strong sense, so there is no Fully Polynomial Time Approximation Scheme (FPTAS) for it unless P = NP [7]; however, a Polynomial Time Approximation Scheme (PTAS) for this problem was designed by Hochbaum and Shmoys [10] and later improved by Azar and Epstein [1].

An efficient polynomial time approximation scheme (EPTAS) for $Q||C_{\max}$ computes a schedule of makespan at most $(1 + \varepsilon)$ times the optimum for any constant $\varepsilon > 0$ in time $f(\varepsilon^{-1})n^{\mathcal{O}(1)}$ for some computable function f. Chen, Jansen, and Zhang [5] show that when all machines have the same speed there is no EPTAS for $Q||C_{\max}$ with runtime smaller than $2^{\mathcal{O}(\varepsilon^{\delta-1})} + n^{\mathcal{O}(1)}$ for any $\delta > 0$ assuming the Exponential Time Hypothesis (ETH). The first EPTAS for $Q||C_{\max}$ with a runtime of $2^{\mathcal{O}(\varepsilon^{-2}\log^3\varepsilon^{-1})} + n^{\mathcal{O}(1)}$ was developed by Jansen [12] and later improved by Jansen, Klein, and Verschae [14] who reduced the runtime to $2^{\mathcal{O}(\varepsilon^{-1}\log^4\varepsilon^{-1})} + n^{\mathcal{O}(1)}$. The previously best known EPTAS for the problem was by Berndt, Brinkop, Jansen, Mnich, and Stamm [2] who designed an algorithm with runtime of $2^{\mathcal{O}(\varepsilon^{-1}\log^3\varepsilon^{-1}\log\log\varepsilon^{-1})} + \mathcal{O}(n)$. For the case when all machines have the same speed (problem denoted as $P||C_{\max}$) the fastest EPTAS for the problem is by Berndt, Deppert, Jansen, and Rohwedder [3] and has runtime $2^{\mathcal{O}(\varepsilon^{-1}\log\varepsilon^{-1}\log\log\varepsilon^{-1})} + \mathcal{O}(n)$. A central part of all the above EPTAS is the formulation of a restricted version of the problem as a Mixed Integer Linear Program (MILP).

1.1 Our Results

Our main result is an improvement on the previously best runtime of an EPTAS for $Q||C_{\max}$:

Theorem 1.1. *There is an EPTAS for $Q||C_{\max}$ with runtime $2^{\mathcal{O}(\varepsilon^{-1}\log^2\varepsilon^{-1})} + \mathcal{O}(n)$.*

This represents a reduction by a factor of $\log\varepsilon^{-1}\log\log\varepsilon^{-1}$ in the exponent of the time complexity of the previously best known EPTAS for the problem, which is a slightly larger reduction in time complexity than that achieved by the algorithm in [2] over the algorithm in [14], and it nearly matches the running time of the best-known EPTAS for $P||C_{\max}$.

Although this is a modest improvement over the running time of the EPTAS in [2], it highlights the difficulty of closing the inapproximability gap of this fundamental problem and brings us closer to the aforementioned currently best known inapproximability bound of the seemingly simpler $P||C_{\max}$.

Our algorithm uses the MILP formulation in [2] for a relaxed version of $Q||C_{\max}$. In this MILP formulation, we only require that a constant number of variables be integer, but the number of constraints involving these variables is $\mathcal{O}(\log n)$. To solve the MILP, we use a divide-and-conquer approach that is similar to that in [16] for solving integer programs, but we require

[1] We write $[n]$ for $\{1, \ldots, n\}$ and $[n]_0$ for $\{0, 1, \ldots, n\}$ for $n \in \mathbb{N}$.

significant new ideas, as the algorithm in [16] only has to deal with integer programs with a constant number of constraints.

First, we repeatedly simplify the MILP by reducing the values of the right-hand sides of the constraints by nearly half through the removal of some jobs and machines, thus decreasing the number of constraints for integer variables until eventually all of them are eliminated. At that point, the MILP transforms into a linear program, which we can solve directly. In the second part of our algorithm we start with the solution of this linear program and repeatedly double the current solution and add back the jobs and machines removed during the first part of the algorithm to create solutions for MILPs with ever larger right-hand sides until we get a solution for the original MILP.

To reduce the right-hand sides of the constraints of the MILP we remove a set of jobs and machines to ensure that (i) the right-hand sides of the constraints are even and hence can be divided by 2, and (ii) the mixed integer program resulting after dividing by 2 the right-hand sides of the constraints has a solution. As we do not know which jobs and machines to remove to guarantee these conditions, we guess them. Guessing the largest jobs and fastest machines to remove might seem relatively easy as there is only a small number of integer variables for them; however, since the machines have different speeds, we need to be careful when defining the notions of "large" and "small" jobs as a job can be large on one machine and small on another.

Another major challenge is that the number of fractional variables could be very large, so we cannot directly guess the set of small jobs to remove from the MILP. A key idea is to instead guess only the total processing time of these small jobs on each machine. The mixed integer program MILP$'$ that we create after dividing the right-hand sides of the constraints by 2 will then have additional constraints reserving space for the small jobs removed, so they can later be added back to the solution.

The algorithm needs to find solutions to a large number of these MILP$'$s, so we use dynamic programming to efficiently compute them. Building a solution for a MILP$'$ from solutions to MILP$'$s with smaller right-hand sides requires selecting the small jobs removed in the first part of the algorithm that will be scheduled on each machine. We formulate this selection problem as a linear program; we show that the solutions to all linear programs needed to schedule the small jobs throughout the entire execution of the algorithm can be stored in a single dynamic programming table.

We note that we do not use an algorithm for directly solving an MILP, but only require an LP solver, different from previous EPTAS approaches where MILP solvers are necessary. This opens the possibility of an efficient implementation of the algorithm with actual practical running time, like that in [3]. We believe that the ideas presented in this paper can be used to find approximate solutions to other problems, such as scheduling jobs with capacity or class constraints [4, 15], scheduling with setup times [13], and bin packing [17].

The rest of the paper is organized as follows. In Section 2 we describe preprocessing steps that simplify the problem and we define the notions of large and small jobs. We formulate the MILP and compute the support of a solution in Section 3. In Section 4 we show how to simplify the MILP by reducing the values on the right-hand sides of the constraints. Section 5 presents our algorithm.

2 Preprocessing and Rounding

A commonly used technique for designing approximation schemes is dual approximation, introduced by Hochbaum and Shmoys [11]. For $Q||C_{\max}$, a dual approximation algorithm with parameter ε has the property that given the value of a possible makespan T the algorithm either outputs a solution with value at most $(1+\varepsilon)T$, or it determines that no solution with value T exists. Combining this approach with binary search yields an approximation scheme. Using dual approximation on $Q||C_{\max}$ turns the problem into a bin packing problem with differently sized bins. Specifically, given a makespan guess T the size of the bin for machine M_i is equal to $T \cdot s_i$. Then a set J of jobs can be scheduled on a machine i if $\sum_{J_j \in J} \frac{p_j}{s_i} \leq T$.

Without loss of generality we may assume $n \geq m$. We define $p_{\max} \coloneqq \max_{i\in[n]} p_i$ and $p_{\min} \coloneqq \min_{i\in[n]} p_i$. We define $s_{\max}$ and $s_{\min}$ analogously. To simplify the problem we perform the same preprocessing and rounding as [2]: Let $\delta > 0$ be a constant such that δ^{-1} is integer and let T^* be the makespan of an optimal schedule. In this optimal schedule we move all jobs of length at most $\frac{\delta p_{\max}}{n}$ to any of the fastest machines; this increases the makespan by at most a factor of δ because these jobs have total processing time at most $\delta p_{\max} \leq \delta T^* s_{\max}$. We also move to a fastest machine all jobs scheduled on the set S of machines of speed at most $\frac{\delta s_{\max}}{n}$. These jobs have total processing time $\sum_{i\in S} T^* s_i \leq \sum_{i\in S} T^* \frac{\delta s_{\max}}{n} \leq nT^* \frac{\delta s_{\max}}{n} = T^* \delta s_{\max}$. Thus, scheduling those jobs on a fastest machine further increases the makespan at most by a factor of δ. After performing the above changes, $p_{\min} > \frac{\delta p_{\max}}{n}$ and $s_{\min} > \frac{\delta s_{\max}}{n}$ which implies $\frac{p_{\min}}{p_{\max}}, \frac{s_{\min}}{s_{\max}} < \frac{\delta}{n}$. Also,

$$s_{\max} \geq \frac{p_{\max}}{T} > \frac{p_{\min}}{T} \geq \frac{\delta p_{\max}}{nT} \geq \frac{\delta s_{\max}}{n^2}. \tag{1}$$

Since $T \leq \frac{np_{\max}}{s_{\max}}$ and $T \geq \frac{p_{\max}}{s_{\max}}$ we use binary search on the interval $[\frac{p_{\max}}{s_{\max}}, \frac{np_{\max}}{s_{\max}}]$ to find the smallest value T for which the algorithm described in the following sections finds a schedule of makespan at most $(1+\varepsilon)T$, for any given constant $\varepsilon > 0$. For each value of T selected in the binary search we divide all job processing times by T; in the sequel p_j denotes the scaled processing time of job J_j. Then, by (1) all processing times and job speeds are in the interval $[\frac{\delta s_{\max}}{n^2}, s_{\max}]$. Let $\kappa \coloneqq \lceil \log(\delta^{-1} n^2) \rceil$, then $2^{-\kappa} s_{\max} = 2^{-\lceil \log(\delta^{-1} n^2) \rceil} s_{\max} \leq 2^{-\log(\delta^{-1} n^2)} s_{\max} = \frac{\delta s_{\max}}{n^2}$, and so all job processing times and machine speeds are in the interval $[2^{-\kappa} s_{\max}, 2^0 s_{\max}]$.

Next, we apply the rounding from [3] to the processing times and machine speeds. Define $b_{k,0} \coloneqq 2^{-k} s_{\max}$ and $b_{k,\ell} \coloneqq b_{k,0}(1 - \frac{\ell\delta}{2})$ for $k \in [\kappa]_0$ and $\ell \in [\delta^{-1}]$. This gives $b_{k-1,\delta^{-1}} = b_{k,0}$ for $k \in [\kappa]$ and $b_{k,\ell-1} \leq (1+\delta) b_{k,\ell}$ for all $k \in [\kappa]_0, \ell \in [\delta^{-1}]$. The total number of values $b_{k,\ell}$ is $\tau \coloneqq \kappa \cdot \delta^{-1} \in \mathcal{O}(\delta^{-1} \log(\delta^{-1} n))$.

For notational simplicity we reindex the $b_{k,\ell}$ values to b_i in non-increasing order, so $b_i = b_{\lfloor \frac{i-1}{\delta^{-1}} \rfloor, i-1 \bmod \delta^{-1}}$. We round every machine speed s_i and processing time p_i in $(b_{r+1}, b_r]$ up to b_r. Since $b_{k,\ell-1} \leq b_{k,\ell}(1+\delta)$ for all $k \in [\kappa]_0, \ell \in [\delta^{-1}]$ this increases the makespan of the schedule by at most a factor of δ.

A *large job* for a machine with speed b_i is a job with processing time in $(\delta b_i, b_i]$. A job is *small* on a machine with speed b_i if its processing time is smaller than or equal to δb_i.

Lemma 2.1. *$\delta b_i \geq b_{i+\delta^{-1}\lceil \log(\delta^{-1}) \rceil}$ for $i \in [\tau]$, so a job is considered large for at most $\delta^{-1} \lceil \log(\delta^{-1}) \rceil$ machine speeds.*

Proof. There are values k, ℓ such that $b_i = s_{\max} \cdot 2^{-k} \cdot (1 - \frac{\ell}{2\delta^{-1}})$. Since $\delta^{-1}\lceil \log(\delta^{-1})\rceil$ is a multiple of δ^{-1}, then

$$b_{i+\delta^{-1}\lceil \log(\delta^{-1})\rceil} = s_{\max} 2^{-k-\lceil \log(\delta^{-1})\rceil}\left(1 - \frac{\ell}{2\delta^{-1}}\right) = b_i \cdot 2^{-\lceil \log(\delta^{-1})\rceil} \leq b_i \cdot 2^{\log(\delta)} = b_i\delta. \quad \square$$

A *configuration* for a machine speed b_i is a collection of large jobs of total processing time at most b_i. More formally, the set $\mathcal{C}_i$ of configurations γ for a machine speed b_i is $\mathcal{C}_i \coloneqq \left\{\gamma \in \mathbb{Z}^{\tau}_{\geq 0} : \gamma_j = 0 \text{ for } b_j \leq \delta b_i, \sum_{j=1}^{\tau} \gamma_j b_j \leq b_i\right\}$ for $i \in [\tau]$. Further, we define $\text{free}(i, \gamma) \coloneqq b_i - \sum_{j=1}^{\tau} \gamma b_j$ for $i \in [\tau], \gamma \in \mathcal{C}_i$ as the space not occupied by large jobs on a machine with speed b_i when the configuration γ is placed on it. The *support* of a vector $v \in \mathbb{R}^k$ is defined as the set of indices of its non-zero components, i.e., $\text{supp}(v) \coloneqq \{i \in [k] : v_i \neq 0\}$.

3 The MILP

We formulate a relaxed version of $Q||C_{\max}$ as the following mixed integer linear program used in [2], in which large jobs are scheduled using configurations (integral and fractional). Configuration variables $x_{i,\gamma}$ indicate how often configuration γ is scheduled on machines with speed b_i. Let $L \coloneqq \delta^{-1}\lceil \log(\delta^{-3} \log \delta^{-1})\rceil$. The MILP is the following.

$$\sum_{\gamma \in \mathcal{C}_i} x_{i,\gamma} = \mu_i \qquad \text{for } i \in [\tau] \tag{c1}$$

$$\sum_{i=1}^{\tau} \sum_{\gamma \in \mathcal{C}_i} \gamma_j x_{i,\gamma} + \sum_{i=1}^{\tau} y_{i,j} = \eta_j \qquad \text{for } j \in [\tau] \tag{c2}$$

$$\sum_{\gamma \in \mathcal{C}_i} \text{free}(i,\gamma) x_{i,\gamma} - \sum_{j=1}^{\tau} b_j y_{i,j} \geq 0 \qquad \text{for } i \in [\tau] \tag{c3}$$

$$x_{i,\gamma}, y_{i,j} \geq 0 \qquad \text{for } i, j \in [\tau] \text{ and } \gamma \in \mathcal{C}_i$$

$$y_{i,j} = 0 \qquad \text{for } i, j \in [\tau] \text{ with } b_j > \delta b_i$$

$$x_{i,\gamma} \in \mathbb{Z}_{\geq 0} \qquad \text{for } i \in [L], \gamma \in \mathcal{C}_i$$

where μ_i is the number of machines with speed b_i, η_i is the number of jobs with processing time b_i, and γ_j is the number of jobs of length b_j in configuration γ. Configuration variables $x_{i,\gamma}$ are integral for the fastest L machine speeds, and fractional for the remaining ones. A variable $y_{i,j}$ specifies the number of jobs of length b_j scheduled on machines of speed b_i. Note that $y_{i,j} > 0$ only if $b_j \leq \delta b_i$, i.e., for jobs with length b_j that are small on machines with speed b_i.

The value of L ensures that the number of fastest machines storing integer configurations is large enough to guarantee that the jobs that need to be removed to reduce the right hand sides of the constraints can use these machines to be added back to the solution and only slightly increasing the makespan. The constraint (c1) ensures that for every machine speed b_i exactly μ_i configurations from $\mathcal{C}_i$ are selected and (c2) guarantees that all jobs are scheduled. The constraint (c3) ensures that the time occupied by small jobs does not exceed the free space left by the configurations.

Building a schedule from a solution to the MILP is described in Lemma 24 of [2]:

Lemma 3.1. *Given a feasible solution of MILP for some makespan T, a schedule with makespan at most $(1+6\delta)T$ can be constructed in time $2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})}+\mathcal{O}(n)$.*

Proof. Integral configuration variables $x_{i,\gamma}$ can be directly turned into a schedule for $x_{i,\gamma}$ machines with speed b_i. The fractional configuration variables need to be handled with more care. For every fractional variable $x_{i,\gamma}$ we can schedule $\lfloor x_{i,\gamma}\rfloor$ machines with speed b_i using configuration γ. Then, if $x_{i,\gamma} > \lfloor x_{i,\gamma}\rfloor$ place a configuration γ on a fastest machine. Let Γ be the set of these configurations placed on a fastest machine.

We need to bound the increase in the makespan caused by placing the configurations in Γ on a fastest machine: Fix a value $h \in \{L+1,\dots,\tau\}$ and write MILP in matrix form as

$$\begin{pmatrix} A & B & C \end{pmatrix}\begin{pmatrix} u \\ w \\ z \end{pmatrix} = b$$

where $z \in \mathbb{Z}^L_{\geq 0}$, $u \in \mathbb{R}^{\beta_h}_{\geq 0}$, and $w \in \mathbb{R}^{\tau-L-\beta_h}_{\geq 0}$ is a solution, A are the columns corresponding to the fractional variables $x_{h\gamma}$, β_h is the number of columns in A, C are the columns corresponding to the integer variables, and B are the columns corresponding to the remaining fractional variables. For vectors w and z, a vertex solution $\hat{u}$ for the following linear program

$$A\hat{u} = b - \begin{pmatrix} B & C \end{pmatrix}\begin{pmatrix} w \\ z \end{pmatrix}$$

has support $\hat{\beta}_h$, where $\hat{\beta}_h$ is the number of constraints for the fractional variables $x_{h\gamma}$. By Lemma 2.1 the number of constraints (c2) in MILP for variables $x_{h\gamma}$ is at most $\delta^{-1}\lceil\log\delta^{-1}\rceil$; also there is one constraint (c1) and one constraint (c3) in MILP for these variables, so $\hat{\beta}_h \leq 2+\lceil\log\delta^{-1}\rceil\delta^{-1}$. Thus, there is a solution for MILP in which for each $h \in \{L+1,\dots,\tau\}$ the number of non-zero fractional variables $x_{h,\gamma}$ is at most $\delta^{-1}\lceil\log\delta^{-1}\rceil+2$ and so the total processing time of the configurations in Γ placed on a fastest machine is

$$\begin{aligned}
(\lceil\log(\delta^{-1})\rceil\delta^{-1}+2)\sum_{k=L+1}^{\tau} b_k &\leq (\lceil\log(\delta^{-1})\rceil\delta^{-1}+2)\sum_{k=L'}^{\infty} b_k \\
&\leq (\lceil\log(\delta^{-1})\rceil\delta^{-1}+2)s_{\max}\sum_{k=\frac{L'}{\delta^{-1}}}^{\infty}\sum_{\ell=0}^{\delta^{-1}-1} 2^{-k}\Big(1-\frac{\ell}{2\delta^{-1}}\Big) \\
&\leq (\lceil\log(\delta^{-1})\rceil\delta^{-1}+2)\delta^{-1}s_{\max}\sum_{k=\frac{L'}{\delta^{-1}}}^{\infty} 2^{-k} \\
&= (\lceil\log(\delta^{-1})\rceil\delta^{-1}+2)\delta^{-1}s_{\max}2^{-\frac{L'}{\delta^{-1}}+1} \\
&= 2(\lceil\log(\delta^{-1})\rceil\delta^{-1}+2)\delta^{-1}s_{\max}2^{-\lceil\log(\delta^{-3}\log(\delta^{-1}))\rceil-\lceil\log\delta^{-2}\rceil+1} \\
&\leq 2(\lceil\log(\delta^{-1})\rceil\delta^{-1}+2)\delta^{-1}s_{\max}2^{-\log(\delta^{-3}\log(\delta^{-1}))}
\end{aligned}$$

$$= 2(\lceil \log(\delta^{-1}) \rceil \delta^{-1} + 2)\delta^{-1} s_{\max} \frac{1}{\delta^{-3} \log(\delta^{-1})}$$
$$= 2\frac{\delta^{-1} \cdot \delta^3 \cdot s_{\max}}{\log(\delta^{-1})}(\lceil \log(\delta^{-1}) \rceil \delta^{-1} + 2) < 5\delta s_{\max}.$$

The last inequality holds for $\delta < \frac{1}{20}$. By Lemma 2.1 there are at most $\delta^{-1} \log \delta^{-1}$ different job lengths that are big on machines with speed b_i. Since at most δ^{-1} jobs are big on a machine with speed b_i then the number of configurations $\mathcal{C}_i$ is $|\mathcal{C}_i| \leq (\delta^{-1} \log \delta^{-1})^{\delta^{-1}} = 2^{\log(\delta^{-1} \log \delta^{-1})\delta^{-1}} \leq 2^{\delta^{-1} \log^2 \delta^{-1}}$. Since the number of different machine speeds is $\delta^{-1} \log(\delta^{-1} n^2)$, then the number of different variables $x_{i,\gamma}$ is at most $2^{\delta^{-1} \log^2(\delta^{-1}} \delta^{-1} \log(\delta^{-1} n^2) = 2^{\delta^{-1} \log^2 \delta^{-1}} 2^{\log \delta^{-1}} \log(\delta^{-1} n^2) = 2^{\mathcal{O}(\delta^{-1} \log^2 \delta^{-1})} \log n$. Hence, scheduling the jobs specified by the configuration variables $x_{i,\gamma}$ can be done in time $2^{\mathcal{O}(\delta^{-1} \log^2 \delta^{-1})} + \mathcal{O}(n)$.

Notice that only complete configurations γ were placed into the schedule and thus potentially more jobs were scheduled than specified by the solution of the MILP because the values for some configuration variables were rounded up. Thus, if needed, we reduce the values of some variables $y_{i,j}$ to remove any extra jobs. After this we have $\sum_{i=1}^{\tau} y_{i,j} \in \mathbb{Z}_{\geq 0}$ for every $j \in [\tau]$.

We sort the small jobs in non-increasing order of processing time and greedily schedule them in the free space of the machines, taking them in non-increasing order of speed, scheduling jobs on a machine until the first job scheduled there exceeds the free space of the machine. This increases the makespan on machines of speed b_i by at most δb_i. Since there are at most $\delta^{-1} \log(\delta^{-1} n^2)$ different jobs sizes and machine speeds, sorting requires time $\mathcal{O}(\delta^{-1} \log^2(\delta^{-1} n))$. Thus, this step requires $\mathcal{O}(n + \delta^{-1} \log^2(\delta^{-1} n)) = \mathcal{O}(n) + \mathcal{O}(\delta^{-1} \log^2 \delta^{-1})$. □

3.1 Solving the MILP: Support Bound

The first step in the solution of MILP is to compute the support of a solution. To this end, we change the constraints (c3) to these

$$\sum_{\gamma \in \mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{b_i \delta} \right\rceil x_{i,\gamma} \geq \sum_{j=1}^{\tau} \frac{b_j}{b_i \delta} y_{i,j} \qquad \text{for } i \in [\tau] \qquad \text{(cr3)}$$

and obtain a new mixed integer program MILPr. For every processing time b_i, $i \in [\tau]$ and configuration $\gamma \in \mathcal{C}_i$, $\|\gamma\|_\infty \leq \delta^{-1}$ as every job in the configuration γ has processing time at least δb_i. Since $\text{free}(i,\gamma) \leq b_i$ for every $i \in [\tau]$ and $\gamma \in \mathcal{C}_i$ then $\lceil \frac{\text{free}(i,\gamma)}{b_i \delta} \rceil \leq \delta^{-1}$. Thus, all the coefficients for the integral variables in MILPr are bounded by δ^{-1}.

Lemma 3.2. *If MILP is feasible, then MILPr is also feasible. Furthermore, from a solution for MILPr we can construct a schedule with a makespan that is at most a factor of $1 + \delta$ longer than the makespan of the schedule for a solution for MILP.*

Proof. Clearly, a solution to MILP also is a solution for MILPr as only the free space in the third constraints of MILP was increased. On the other hand, the modification of the free space in the constraints (cr3) of MILPr increases the makespan of the schedule corresponding to a solution for MILP by at most a factor of $1 + \delta$ because $\text{free}(i,\gamma) + \delta b_i \geq \delta b_i \lceil \frac{\text{free}(i,\gamma)}{\delta b_i} \rceil$. □

The support of solutions for MILPr can be bounded by using a result of Eisenbrand and Shmonin [6]:

Lemma 3.3. *If MILPr has a solution, then there is a solution to MILPr where the support of the integral variables $x_{i,\gamma}$ is bounded by $\mathcal{O}(\delta^{-1}\log^2\delta^{-1})$.*

Proof. There are at most $4L$ constraints in MILPr concerning integral variables: L for constraints (c1), $L+\delta^{-1}\lceil\log\delta^{-1}\rceil < 2L$ for the constraints (c2) because by Lemma 2.1 a job is big in at most $\delta^{-1}\lceil\log\delta^{-1}\rceil$ machines, and L for the constraints (cr3). Eisenbrand and Shmonin [6] showed that if a mixed integer program has a solution, then it has a solution with support for the integer variables bounded by $2c\log(4c\Delta)$, where c is the number of constraints and Δ is biggest absolute value of an entry in the constraint matrix. Since L is $\mathcal{O}(\delta^{-1}\log\delta^{-1})$, there is a solution for MILPr for which the support for the integral variables is

$$\begin{aligned} 2(4L)\log(4(4L)\delta^{-1}) &= \mathcal{O}(\delta^{-1}\log\delta^{-1}\log(\delta^{-2}\log\delta^{-1})) \\ &= \mathcal{O}(\delta^{-1}\log\delta^{-1}\log\delta^{-2}) \\ &= \mathcal{O}(\delta^{-1}\log^2\delta^{-1}). \end{aligned}$$

□

4 Halving The Right Hand-Sides of the Constraints

If MILP has a solution (x,y) corresponding to a schedule of makespan T, our algorithm computes a schedule of makespan at most $T(1+\mathcal{O}(\delta))$. We use a divide-and-conquer approach that simplifies MILP by reducing the right-hand sides of the constraints to nearly half their values. This requires that we first remove some jobs and machines to ensure that the right-hand sides of the constraints are even.

Some of the machines removed might store fractions of jobs that belong to configurations; to ensure that we can later schedule these fractional jobs as whole jobs, we need to create a gap between the largest processing time of these jobs and the L-th fastest machine speed. To achieve this, first we handle the configuration variables $x_{i,\gamma}$ with $L < i \le L' := L+\delta^{-1}\lceil\log\delta^{-2}\rceil$ as integer and later we allow them to take fractional values; this creates the desired gap of size $\delta^{-1}\lceil\log\delta^{-1}\rceil$.

Consider a feasible instance of MILP, for which $\mu_i > 0$ for at least one $i \in [L]$, and the corresponding MILPr. Let (x,y) be a solution to MILPr with small support, which exists by Lemma 3.3. We show how to construct a new mixed integer program MILP′ from MILPr where the right-hand sides (μ',η') of the constraints have values close to one-half of their initial values and for which there is a solution (x',y') with small support.

To ensure that MILP′ has a solution, we construct it so that for all $i \in [L']$, $x'_{i,\gamma} = \lfloor\frac{x_{i,\gamma}}{2}\rfloor$ is the integer part of a solution if $x_{i,\gamma}$ is the integer part of a solution for MILPr . Note that by constraint (c1) rounding down $\frac{x_{i,\gamma}}{2}$ might require removing some of the machines of the L' fastest speeds. We will denote the number of these removed machines using variables c_i. Similarly, the rounding will cause some of the jobs in the configurations for these rounded variables $x_{i,\gamma}$ to be removed; we will denote the number of these removed jobs with variables d_j.

For $i \in [L']$ let $x'_{i,\gamma} = \lfloor \frac{x_{i,\gamma}}{2} \rfloor$ and $x''_{i,\gamma} = x_{i,\gamma} - 2x'_{i,\gamma} \in \{0,1\}$ (hence, $x''_{i,\gamma} = 1$ iff $x_{i,\gamma}$ is odd). For $i > L'$, let $x'_{i,\gamma} = \frac{x_{i,\gamma}}{2}$. Now we compute the values for μ' and η'.

(I) First we determine μ'. For each $i \in [\tau]$, if $\mu_i = 0$ we set $\mu'_i = 0$; otherwise, we consider two cases:

- For $i \in [L']$, let $c_i = \sum_{\gamma \in \mathcal{C}_i} x''_{i,\gamma}$. If $c_i = 0$ for all $i \in [L]$ then we select a variable $x_{h,\gamma} \geq 2$ for some $h \in [L]$ (this variable exists because $\mu_i > 0$ for at least one $i \in [L]$) and set $c_h = 2$, $x''_{h,\gamma} = 2$, and $x'_{h,\gamma} = \frac{x_{h,\gamma}-2}{2} \in \mathbb{Z}_{\geq 0}$. As we explain in Section 5.1, we need that $c_h > 0$ for at least one $h \in [L']$ so that at least one machine of speed s_h, $h \in [L']$ is removed; that machine is needed to schedule some of the small jobs that will be removed in Steps (II) and (III) to create MILP′.

  For all $i \in [L']$ we then set

  $$\mu'_i := \sum_{\gamma \in C_i} x'_{i,\gamma} = \sum_{\gamma \in C_i} \frac{x_{i,\gamma} - x''_{i,\gamma}}{2} = \frac{\mu_i - c_i}{2} \in \mathbb{Z}_{\geq 0}.$$

  Note that if $\mu'_i < \mu_i/2$ we remove some machines from MILPr before dividing by 2 the right hand sides of its constraints to create MILP′. By removing machines we also remove time slots where small jobs were scheduled in the solution for MILPr. To obtain a feasible solution for the original MILP we later introduce variables Free_i to keep track of these lost time slots for small jobs.
- For each $i > L'$ we set $\mu'_i = \frac{\mu_i}{2}$.

We now determine η'. For each $j \in [\tau]$, if $\eta_j = 0$ we set $\eta'_j = 0$; otherwise, we specify the value of η'_j in Step (II) for $j \in [L']$ and in Step (III) for $j > L'$.

Let $L'' \in [\tau]$ be such that $b_{L''} \geq \delta b_{L'}$ and $b_{L''+1} < \delta b_{L'}$, i.e., $b_{L''}$ is the smallest job processing time that is considered big for the machine speed $b_{L'}$.

(II) For each $j \in [L']$ we consider two cases.

- If $b_j > \delta s_{\max}$, then a job of length b_j is not small on any machine, so we set

  $$\eta'_j := \sum_{i=1}^{j} \sum_{\gamma \in \mathcal{C}_i} \gamma_j x'_{i,\gamma} = \sum_{i=1}^{j} \sum_{\gamma \in C_i} \gamma_j \frac{x_{i,\gamma} - x''_{i,\gamma}}{2} = \frac{\eta_j - d_j}{2} \in \mathbb{Z}_{\geq 0}$$

  where $d_j = \sum_{i=1}^{j} \sum_{\gamma \in \mathcal{C}_i} \gamma_j x''_{i,\gamma}$. Note that $\eta'_j \in \mathbb{Z}_{\geq 0}$ as $x'_{i,\gamma} \in \mathbb{Z}_{\geq 0}$.
- If $b_j \leq \delta s_{\max}$, let s_k be the slowest speed that is at least $\delta^{-1} b_j$; then a job of length b_j is small on machines with speed at least s_k. Note that $s_k = b_k < b_j$, and thus

  $$\sum_{i=1}^{k} y_{i,j} + \sum_{i=k+1}^{j} \sum_{\gamma \in \mathcal{C}_i} 2\gamma_j x'_{i,\gamma} = \sum_{i=1}^{k} y_{i,j} + \sum_{i=k+1}^{j} \sum_{\gamma \in \mathcal{C}_i} \gamma_j (x_{i,\gamma} - x''_{i,\gamma}) = \eta_j - d_j$$

  where $d_j = \sum_{i=k+1}^{j} \sum_{\gamma \in \mathcal{C}_i} \gamma_j x''_{i,\gamma}$. We know that d_j is integer as $x''_{i,\gamma}$ is integer; thus, the left-hand side of the equation is integral and particularly $\sum_{i=1}^{k} y_{i,j}$ is

integral. Furthermore, if $\eta_j - d_j$ is odd then $\sum_{i=1}^{k} y_{i,j}$ must be odd and at least 1. We select the smallest possible values $y''_{i,j} \leq y_{i,j}$ for all $i \in [k]$ so that $\sum_{i=1}^{k}(y_{i,j} - y''_{i,j})$ is even and define $y'_{i,j} = \frac{y_{i,j} - y''_{i,j}}{2}$. We then set

$$\eta'_j := \sum_{i=1}^{k} y'_{i,j} + \sum_{i=k+1}^{j} \sum_{\gamma \in \mathcal{C}_i} \gamma_j x'_{i,\gamma} = \left\lfloor \frac{\eta_j - d_j}{2} \right\rfloor.$$

Note that $\sum_{i=1}^{\tau} y''_{i,j} \in \{0, 1\}$. Observe also that the new mixed integer program where the right-hand sides of the constraints are the η'_j values does not consider the fractions of the jobs corresponding to the values $y''_{i,j}$. To be able to schedule these discarded jobs we need to allocate space in some of the machines for them. To this end we will introduce variables Rsrvd_i, which we explain later.

(III) For $j > L'$ we consider two cases. Let $s_{k+1} = b_{k+1}$ be the fastest speed such that $b_j \geq \delta b_{k+1}$.

- If $k + 1 \leq L'$, so $j \leq L''$, then

$$\sum_{i=1}^{k} y_{i,j} + \sum_{i=k+1}^{L'} \sum_{\gamma \in \mathcal{C}_i} 2\gamma_j x'_{i,\gamma} + \sum_{i=L'+1}^{j} \sum_{\gamma \in \mathcal{C}_i} \gamma_j x_{i,\gamma} = \eta_j - d_j$$

where $d_j = \sum_{i=k+1}^{L'} \sum_{\gamma \in \mathcal{C}_i} \gamma_j x''_{i,\gamma}$. Note that $\sum_{i=1}^{k} y_{i,j} + \sum_{i=L'+1}^{j} \sum_{\gamma \in \mathcal{C}_i} \gamma_j x_{i,\gamma}$ is integral and it is odd if $\eta_j - d_j$ is odd because $\sum_{i=k+1}^{L'} \sum_{\gamma \in \mathcal{C}_i} 2\gamma_j x'_{i,\gamma}$ is even. To make the right-hand side even we cannot just decrease the variables $y_{i,j}$ as $\sum_{i=1}^{k} y_{i,j}$ may be smaller than 1. Instead, we decrease the variables $y_{i,j}$ and if needed we adjust the configuration variables $x_{i,\gamma}$ for $i > L'$ by removing the smallest number of fractional jobs of length b_j from them to achieve this: Specifically, if a configuration γ contains a job of length b_j, we decrease the value $x_{i,\gamma}$ for that configuration and increase accordingly the value $x_{i,\gamma'}$ of a configuration variable for which γ' is obtained from γ by decreasing γ_j by one.

We then create variables $x'_{i,\gamma}$ and $y'_{i,j}$ such that

$$\left\lfloor \frac{1}{2} \left(\sum_{i=1}^{k} y_{i,j} + \sum_{i=L'+1}^{j} \sum_{\gamma \in \mathcal{C}_i} \gamma_j x_{i,\gamma} \right) \right\rfloor = \sum_{i=1}^{k} y'_{i,j} + \sum_{i=L'+1}^{j} \sum_{\gamma \in \mathcal{C}_i} \gamma_j x'_{i,\gamma},$$

$2y'_{i,j} \leq y_{i,j}$ and $\sum_{\gamma \in \mathcal{C}_i} 2x'_{i,\gamma} = \mu_i$ for every $i > L'$. Finally, we set

$$\eta'_j := \sum_{i=1}^{k} y'_{i,j} + \sum_{i=k+1}^{j} \sum_{\gamma \in \mathcal{C}_i} \gamma_j x'_{i,\gamma} = \left\lfloor \frac{\eta_j - d_j}{2} \right\rfloor.$$

Let $y''_{i,j} = y_{i,j} - 2y'_{i,j}$ for all $i \in [L']$; this is the number of jobs of length b_j removed from machines of speed b_i.

- Otherwise, $b_j < \delta b_{L'}$, and so no integral configuration variable $x_{i,\gamma}$ has $b_j \in \gamma$; thus, we can simply set $y'_{i,j} := \frac{y_{i,j}}{2}$, $x'_{i,\gamma} := \frac{x_{i,\gamma}}{2}$, and

$$\eta'_j := \sum_{i=1}^{\tau} y'_{i,j} + \sum_{i=L'+1}^{j} \sum_{\gamma \in \mathcal{C}_i} \gamma_j x'_{i,\gamma} = \frac{\eta_j}{2}.$$

The above description suggests that the variables $x'_{i,\gamma}$ and $y'_{i,j}$ are determined one at a time, but this might not be the case for the fractional configuration variables as they may be adjusted several times, which the notation does not reflect.

4.1 MILP'

The values μ'_i for $i \in [L']$ computed in Step (I) could be such that $\mu_i > 2\mu'_i$, which means that some machines of speed b_i and the corresponding configurations $\gamma \in \mathcal{C}_i$ scheduled on them by a solution (x, y) of MILPr are removed and not considered in MILP′. Similarly, for some of the values η'_j computed in Steps (II) and (III) it might be that $\eta_j > 2\eta'_j$, which means that some jobs need to be removed from MILPr to create MILP′.

We do not know the values of the variables c_i and d_j in Steps (I) to (III), or the sets of jobs and machines that need to be removed from MILPr to construct MILP′. Instead of trying to determine the number of small jobs removed in Step (I) when some machines and their configurations are discarded (this information is needed to be able to construct a schedule for MILP from a schedule for MILP′), we instead specify the amount of time that these jobs occupy on machines of speed b_i by using variables Free_i for all $i \in [L']$. To reduce the number of possible values for these variables we round up the space occupied by the removed jobs on machines of speed b_i to the nearest multiple of δb_i. This will increase the length of the schedule by at most a factor of δ. Hence, we define

$$\text{Free}_i := \left\lceil \frac{1}{2} \sum_{\gamma \in \mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{\delta b_i} \right\rceil x''_{i,\gamma} \right\rceil + 1 \tag{2}$$

the term $+1$ is needed to ensure feasibility of MILP′.

Similarly, we use variables Rsrvd_i to specify the amount of time needed to schedule on machines of speed b_i, $i \in [L']$, the jobs removed when the values of the variables $y_{i,j}$ are decreased in Steps (II) and (III). As above, to reduce the number of possible values for these amounts of time we round them up to the nearest multiple of δb_i. This rounding further increases the length of the schedule by at most a factor of δ. Thus, we define

$$\text{Rsrvd}_i := \left\lceil \frac{1}{2} \left\lceil \sum_{j=1}^{L'} \frac{b_j}{\delta b_i} y''_{i,j} \right\rceil \right\rceil. \tag{3}$$

We show in the proof of Lemma 5.4 how to schedule the jobs removed when the variables $x_{i,\gamma}$ are decreased in Step (III). Now we can specify the new mixed integer program MILP′:

(MILP′)

$$\sum_{\gamma\in\mathcal{C}_i} x_{i,\gamma} = \mu'_i = \frac{\mu_i - c_i}{2} \qquad \text{for } i \in [L']$$

$$\sum_{\gamma\in\mathcal{C}_i} x_{i,\gamma} = \mu'_i = \frac{\mu_i}{2} \qquad \text{for } i \in \{L'+1,\dots,\tau\}$$

$$\sum_{i=1}^{\tau}\sum_{\gamma\in\mathcal{C}_i} \gamma_j x_{i,\gamma} + \sum_{i=1}^{\tau} y_{i,j} = \eta'_j = \left\lfloor \frac{\eta_j - d_j}{2} \right\rfloor \qquad \text{for } j \in [L'']$$

$$\sum_{i=1}^{\tau}\sum_{\gamma\in\mathcal{C}_i} \gamma_j x_{i,\gamma} + \sum_{i=1}^{\tau} y_{i,j} = \eta'_j = \frac{\eta_j}{2} \qquad \text{for } j \in \{L''+1,\dots,\tau\}$$

$$\sum_{\gamma\in\mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{b_i\delta} \right\rceil x_{i,\gamma} \geq \sum_{j=1}^{\tau} \frac{b_j}{b_i\delta} y_{i,j} + \text{Rsrvd}_i - \text{Free}_i \qquad \text{for } i \in [\tau]$$

$$x_{i,\gamma}, y_{i,j} \geq 0 \qquad \text{for } i,j \in [\tau], \gamma \in \mathcal{C}_i$$

$$y_{i,j} = 0 \qquad \text{for } i,j \in [\tau], b_j > \delta b_i$$

$$x_{i,\gamma} \in \mathbb{Z}_{\geq 0} \qquad \text{for } i \in [L], \gamma \in \mathcal{C}_i$$

Note that $\text{Free}_i = \text{Rsrvd}_i = 0$ for every $i > L'$. The following lemma shows that given a feasible instance of MILPr there is a corresponding feasible mixed-integer program MILP$'$ in which the right-hand sides of the constraints have nearly half the values of the constraints in MILPr and for which there is a solution with small support for the integer variables.

Lemma 4.1. *Given a feasible instance of MILPr where $\mu_i > 0$ for some $i \in [L]$ and in which the support of the integral variables of a solution (x, y) is bounded by s, then there is a feasible mixed integer program* MILP$'$ *in which the right-hand sides $(\mu', \eta', \text{Free}, \text{Rsrvd})$ of the constraints are such that*

1. *$\mu_i - s \leq 2\mu'_i \leq \mu_i$ for $i \in [L']$ and $2\mu'_i = \mu_i$ for $i > L'$,*
2. *$2\mu'_i < \mu_i$ for at least one $i \in [L]$,*
3. *$\eta_j - s\delta^{-1} - 1 \leq 2\eta'_j \leq \eta_j$ for $j \in [L'']$ and $2\eta'_j = \eta_j$ for $j > L''$,*
4. *$\text{Free}_i \leq s\delta^{-1} + 1$ for $i \in [L']$, and*
5. *$\text{Rsrvd}_i \leq L'$ for $i \in [L']$.*

Furthermore, there is a solution for MILP$'$ *for which the support of the integral variables is bounded by s.*

Proof. From the solution (x, y) define x', x'', y', and y'' as in Steps (I) to (III). To prove that MILP$'$ is feasible, we show that (x', y') is a solution. Observe that the first 4 constraints of MILP$'$ are satisfied by the solution by defining μ' and η' as specified in Steps (I) to (III). To show that the fifth constraint is satisfied, let $i \in [L']$ (the constraint is trivially satisfied for $i > L'$ as then $\text{Free}_i = \text{Rsrvd}_i = 0$). Then, by (2) and (3)

$$2\Big(\text{Free}_i + \sum_{\gamma\in\mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{b_i\delta} \right\rceil x'_{i,\gamma}\Big) = 2\left\lceil \frac{1}{2} \sum_{\gamma\in\mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{\delta b_i} \right\rceil x''_{i,\gamma} \right\rceil + 2 + 2\sum_{\gamma\in\mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{b_i\delta} \right\rceil x'_{i,\gamma}$$

$$
\begin{aligned}
&\geq 2 + \sum_{\gamma \in \mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{\delta b_i} \right\rceil (2x'_{i,\gamma} + x''_{i,\gamma}) \\
&= 2 + \sum_{\gamma \in \mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{b_i \delta} \right\rceil x_{i,\gamma} \geq 2 + \sum_{j=1}^{\tau} \frac{b_j}{b_i \delta} y_{i,j} \\
&= 2 + \sum_{j=1}^{\tau} \frac{b_j}{b_i \delta} (2y'_{i,j} + y''_{i,j}) \\
&\geq 1 + 2 \sum_{j=1}^{\tau} \frac{b_j}{b_i \delta} y'_{i,j} + \left\lceil \sum_{j=1}^{\tau} \frac{b_j}{b_i \delta} y''_{i,j} \right\rceil \\
&\geq 2 \sum_{j=1}^{\tau} \frac{b_j}{b_i \delta} y'_{i,j} + 2 \left\lceil \frac{1}{2} \left\lceil \sum_{j=1}^{\tau} \frac{b_j}{b_i \delta} y''_{i,j} \right\rceil \right\rceil \\
&= 2 \Big(\sum_{j=1}^{\tau} \frac{b_j}{b_i \delta} y'_{i,j} + \text{Rsrvd}_i \Big)
\end{aligned}
$$

which implies $\text{Free}_i + \sum_{\gamma \in \mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{b_i \delta} \right\rceil x'_{i,\gamma} \geq \sum_{j=1}^{\tau} \frac{b_j}{b_i \delta} y'_{i,j} + \text{Rsrvd}_i$.

Thus, what remains is to show the other properties stated in the lemma.

1. The bounds for μ'_i follow from Step (I) as $0 \leq c_i \leq s$ for $i \in [L']$.
2. Follows from Step (I) as $c_h > 0$.
3. In Steps (II) and (III) some jobs are removed to make even the right hand sides of the constraints of MILP for the integer configurations variables. Since every job size can appear at most δ^{-1} times in a configuration and the number of integer variables is s, the number of jobs removed from integer configurations specified by the variables d_j for $j \in [L'']$ is at most $s\delta^{-1}$. A single additional job might have to be removed in Step (III) to make even the right-hand sides of the constraints. For $j > L''$, $2\eta'_j = \eta_j$ as stated in Step (III).
4. The number of integral configurations $x_{i,\gamma}$ removed in Step (I) to make the number of machines $\mu_i - c_i$ even is at most s; hence, since for each configuration $\gamma \in \mathcal{C}_i$, $i \in [L']$, $\left\lceil \frac{\text{free}(i,\gamma)}{\delta b_i} \right\rceil \leq \left\lceil \frac{b_i}{\delta b_i} \right\rceil = \delta^{-1}$, then $\text{Free}_i \leq s\delta^{-1} + 1$, the $+1$ term comes from (2).
5. Step (II) makes the right-hand sides of the MILP constraints even by removing at most one job for every size b_j, $j \in [L']$. Since every removed job increases the value for Rsrvd_i by at most one, the claim follows.

Regarding the support of (x', y'), note that the number of integral variables does not increase in Steps (I) to (III). □

4.2 Iterating the Halving Step

We have shown that if MILP is feasible then MILP′ is also feasible for some values of μ' and η' close to half the values of μ and η. Now we repeatedly apply the same approach and reduce

by nearly half the right-hand sides of the constraints of MILP′ to get a series of mixed integer programs with ever smaller right-hand sides to the point where all constraints for the integer variables have been eliminated. At that moment note that the mixed integer program transforms into a linear program.

Let MILP′$(\mu, \eta, \text{Free}, \text{Rsrvd})$ be the instance of the mixed integer program MILP′ when the right-hand sides of the constraints are the vectors μ, η, Free, and Rsrvd. To be able to repeatedly reduce the right-hand sides of the constraints of MILP′ we use this result:

Corollary 4.1.1. *If* MILP′$(\mu, \eta, \text{Free}, \text{Rsrvd})$ *with* $\mu_i > 0$ *for some* $i \in [L]$ *has a solution with* s *non-zero integral variables, then there are vectors* $\mu', \eta', \text{Free}', \text{Rsrvd}'$ *for which* MILP′$(\mu', \eta', \text{Free}', \text{Rsrvd}')$ *has a solution, and such that*

1. $\mu_i - s \leq 2\mu'_i \leq \mu_i$ *for* $i \in [L']$ *and* $2\mu'_i = \mu_i$ *for* $i > L'$,
2. $2\mu'_i < \mu_i$ *for at least one* $i \in [L]$,
3. $\eta_j - (s+1)\delta^{-1} - 1 \leq 2\eta'_j \leq \eta_j$ *for* $j \in [L'']$ *and* $2\eta'_j = \eta_j$ *for* $j > L''$,
4. *the number of non-zero integral variables in the solution is bounded by* s,
5. $\text{Free}_i \leq 2\text{Free}'_i \leq \text{Free}_i + (s+1)\delta^{-1} + 2$ *for* $i \in [L']$, *and*
6. $\text{Rsrvd}_i \leq 2\text{Rsrvd}'_i \leq \text{Rsrvd}_i + L' + 1$ *for* $i \in [L']$.

Proof. Given a solution (x, y) to MILP′ $(\mu, \eta, \text{Free}, \text{Rsrvd})$ with s non-zero integral variables, define x', x'', y', and y'' as in Steps (I) to (III). We show that (x', y') is a solution for MILP′$(\mu', \eta', \text{Free}', \text{Rsrvd}')$, where

$$\text{Free}'_i := \left\lceil \frac{1}{2}\left(\text{Free}_i + \sum_{\gamma \in \mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{\delta b_i} \right\rceil x''_{i,\gamma}\right) \right\rceil + 1 \tag{4}$$

and

$$\text{Rsrvd}'_i := \left\lceil \frac{1}{2}\left(\text{Rsrvd}_i + \left\lceil \sum_{j=1}^{L'} \frac{b_j}{\delta b_i} y''_{i,j} \right\rceil\right) \right\rceil. \tag{5}$$

The first 4 constraints are satisfiable by defining μ' and η' as specified in Steps (I) to (III). For the fifth constraint let $i \in [L']$, then

$$\begin{aligned}
&2\left(\text{Free}'_i + \sum_{\gamma \in \mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{b_i \delta} \right\rceil x'_{i,\gamma}\right) \\
&\quad = 2\left\lceil \frac{1}{2}\left(\text{Free}_i + \sum_{\gamma \in \mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{\delta b_i} \right\rceil x''_{i,\gamma}\right) \right\rceil + 2 + 2\sum_{\gamma \in \mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{b_i \delta} \right\rceil x'_{i,\gamma} \\
&\quad \geq \text{Free}_i + \sum_{\gamma \in \mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{\delta b_i} \right\rceil x''_{i,\gamma} + 2 + 2\sum_{\gamma \in \mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{b_i \delta} \right\rceil x'_{i,\gamma} \\
&\quad \geq 2 + \text{Free}_i + \sum_{\gamma \in \mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{\delta b_i} \right\rceil (2x'_{i,\gamma} + x''_{i,\gamma}) = 2 + \text{Free}_i + \sum_{\gamma \in \mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{b_i \delta} \right\rceil x_{i,\gamma}
\end{aligned}$$

$$\geq 2 + \sum_{j=1}^{\tau} \frac{b_j}{b_i\delta} y_{i,j} + \text{Rsrvd}_i = 2 + \sum_{j=1}^{\tau} \frac{b_j}{b_i\delta}(2y'_{i,j} + y''_{i,j}) + \text{Rsrvd}_i$$
$$\geq 2\sum_{j=1}^{\tau} \frac{b_j}{b_i\delta} y'_{i,j} + 2\left\lceil \frac{1}{2}\left(\text{Rsrvd}_i + \left\lceil \sum_{j=1}^{\tau} \frac{b_j}{b_i\delta} y''_{i,j} \right\rceil\right)\right\rceil = 2\left(\sum_{j=1}^{\tau} \frac{b_j}{b_i\delta} y'_{i,j} + \text{Rsrvd}'_i\right),$$

which implies $\text{Free}'_i + \sum_{\gamma\in\mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{b_i\delta} \right\rceil x'_{i,\gamma} \geq \sum_{j=1}^{\tau} \frac{b_j}{b_i\delta} y'_{i,j} + \text{Rsrvd}'_i$.

1-3. The proofs for the first three properties stated in the lemma and the bound for the number of non-zero integer variables are the same as those for Lemma 4.1.

5. Follows from (4) and the fourth property of Lemma 4.1.

6. Follows from (5) and the fifth property of Lemma 4.1. □

By repeated application of Corollary 4.1.1 we can obtain a sequence of vectors

$$(\mu^{(0)}, \eta^{(0)}, \text{Free}^{(0)}, \text{Rsrvd}^{(0)}), (\mu^{(1)}, \eta^{(1)}, \text{Free}^{(1)}, \text{Rsrvd}^{(1)}), \ldots, (\mu^{(\rho)}, \eta^{(\rho)}, \text{Free}^{(\rho)}, \text{Rsrvd}^{(\rho)})$$

such that if $\text{MILP}'(\mu^{(0)}, \eta^{(0)}, \text{Free}^{(0)}, \text{Rsrvd}^{(0)})$ has a solution, then for every $\ell \leq \rho$, mixed-integer program $\text{MILP}'(\mu^{(\ell)}, \eta^{(\ell)}, \text{Free}^{(\ell)}, \text{Rsrvd}^{(\ell)})$ also has a solution. Furthermore, there exists $i \in [L]$ such that $\mu_i^{(\rho-1)} > 0$ and $\mu_j^{(\rho)} = 0$ for every $j \in [L]$. Note that ρ is $\mathcal{O}(\log m)$ because the number of machines in the constraints for integral variables is reduced by at least half with every application of Corollary 4.1.1. The following corollary bounds the sets of possible values for the vectors $\mu^{(\ell)}$, $\eta^{(\ell)}$, $\text{Free}^{(\ell)}$, and $\text{Rsrvd}^{(\ell)}$.

Corollary 4.1.2. *Let s be the number non-zero integral variables of a solution for MILPr. Then for every $\ell \in [\rho]$*

1. $\frac{\mu_i^{(0)}}{2^\ell} - s \leq \mu_i^{(\ell)} \leq \frac{\mu_i^{(0)}}{2^\ell}$ *for every* $i \in [L']$,
2. $\frac{\eta_j^{(0)}}{2^\ell} - (s+1)\delta^{-1} - 1 \leq \eta_j^{(\ell)} \leq \frac{\eta_j^{(0)}}{2^\ell}$ *for every* $j \in [L'']$,
3. $0 \leq \text{Free}_i^{(\ell)} \leq \sum_{\ell'=1}^{\ell} \frac{(s+1)\delta^{-1}+2}{2^{\ell'}} \leq (s+1)\delta^{-1} + 2$ *for every* $i \in [L']$, *and*
4. $0 \leq \text{Rsrvd}_i^{(\ell)} \leq \sum_{\ell'=1}^{\ell} \frac{L'+1}{2^{\ell'}} \leq L' + 1$ *for every* $i \in [L']$.

Proof.

1. For $\ell = 1$, $\frac{\mu_i^{(0)}}{2} - \frac{s}{2} \leq \mu_i^{(1)} \leq \frac{\mu_i^{(0)}}{2}$ from Corollary 4.1.1. For $\ell > 1$:

$$\frac{\mu_i^{(0)}}{2^\ell} - s \leq \frac{\mu_i^{(0)}}{2^\ell} - \sum_{\ell'=1}^{\ell} \frac{s}{2^{\ell'}} = \frac{1}{2}\left(\frac{\mu_i^{(0)}}{2^{\ell-1}} - \sum_{\ell'=1}^{\ell} \frac{s}{2^{\ell'-1}}\right) = \frac{1}{2}\left(\frac{\mu_i^{(0)}}{2^{\ell-1}} - s - \sum_{\ell'=1}^{\ell-1} \frac{s}{2^{\ell'}}\right)$$
$$\leq \frac{1}{2}\left(\mu_i^{(\ell-1)} - s\right) \leq \frac{1}{2}\left(2\mu_i^{(\ell)} + s - s\right) = \mu_i^{(\ell)}$$

$$\leq \frac{1}{2}\mu_i^{(\ell-1)} \leq \frac{1}{2}\frac{\mu_i^{(0)}}{2^{\ell-1}} = \frac{\mu_i^{(0)}}{2^\ell}.$$

The last 4 inequalities follow from Corollary 4.1.1.

2. The proof is the same as above, but replacing μ_i with η_j and s with $s\delta^{-1} - 1$.

3. For $\ell = 1$, $\text{Free}_i^{(1)} \leq \frac{s\delta^{-1}+1}{2} \leq s\delta^{-1}+1$ from Corollary 4.1.1. For $\ell > 1$, we use induction and Corollary 4.1.1:

$$\begin{aligned}
\text{Free}_i^{(\ell)} &\leq \frac{1}{2}(\text{Free}_i^{(\ell-1)} + s\delta^{-1} + 1) \\
&\leq \frac{1}{2}\Big(\sum_{\ell'=1}^{\ell-1} \frac{s\delta^{-1}+1}{2^{\ell'}} + s\delta^{-1} + 1\Big) \text{ by induction hypothesis} \\
&\leq \sum_{\ell'=1}^{\ell-1} \frac{s\delta^{-1}+1}{2^{\ell'+1}} + \frac{s\delta^{-1}+1}{2} \\
&\leq \sum_{\ell'=1}^{\ell} \frac{s\delta^{-1}+1}{2^{\ell'}} \\
&\leq s\delta^{-1} + 1.
\end{aligned}$$

4. The proof is the same as above, but replacing $s\delta^{-1} + 1$ with L'. □

5 The Dynamic Program

To solve MILP we construct a dynamic programming table $\mathcal{T}$ with $\mathcal{O}(\log m)$ entries, where each entry corresponds to one of the values ℓ in Corollary 4.1.2. Each entry $\mathcal{T}[\ell]$ of the table will store a collection of tuples of the form $(\mu, \eta, \text{Free}, \text{Rsrvd})$, where

- $\mu \in \mathbb{Q}^\tau$ is a vector with the number of machines of each speed, where $\mu_i \in \mathbb{N}$ for $i \leq L'$
- $\eta \in \mathbb{Q}^\tau$ is a vector with the number of jobs of each length, where $\eta_i \in \mathbb{N}$ for $i \leq L''$
- $\text{Free} \in \mathbb{N}^\tau$ is a vector with the values Free_i of Corollary 4.1.1, where $\text{Free}_i = 0$ for $i > L'$
- $\text{Rsrvd} \in \mathbb{N}^\tau$ is a vector with the values Rsrvd_i of Corollary 4.1.1, where $\text{Rsrvd}_i = 0$ for $i > L'$.

The easiest way to think about our algorithm to solve MILP is that it works in three stages. In the first stage it processes the table $\mathcal{T}$ top-down, from row $\ell = 0$ to row $\ell = k$, computing all possible values for $\mu^{(\ell)}$, $\eta^{(\ell)}$, $\text{Free}^{(\ell)}$, and $\text{Rsrvd}^{(\ell)}$, as stated in Corollary 4.1.2. The first entry of the table has a single tuple $(\mu, \eta, \text{Free}, \text{Rsrvd})$, where $\mu = (\mu_1, \mu_2, \ldots, \mu_\tau)$ contains the initial number μ_i of machines for each $i \in [\tau]$, $\eta = (\eta_1, \eta_2, \ldots, \eta_\tau)$ is the initial number of jobs, $\text{Free} = (0, 0, \ldots, 0) \in \mathbb{Z}_{\geq 0}^{|\tau|}$, and $\text{Rsrvd} = (0, 0, \ldots, 0) \in \mathbb{Z}_{\geq 0}^{|\tau|}$.

Corollary 4.1.2 allows us to bound the number of tuples in $\mathcal{T}$. For each row $\mathcal{T}[\ell]$ of the table the number of different values for μ_i ($i \in [L']$) is at most $s+1$ (where s is $\mathcal{O}(\delta^{-1}\log^2\delta^{-1})$ as stated in lemma 3.3), the number of different values for η_j ($j \in [L'']$) is at most $(s+1)\delta^{-1}+2$, the number of different values for Free_i ($i \in [L']$) is at most $(s+1)\lceil\delta^{-1}\rceil+3$, and the number of different values for Rsrvd_i ($i \in [L']$) is at most $L'+2$. Since L' and L'' are $\mathcal{O}(\delta^{-1}\log\delta^{-1})$ and s is $\mathcal{O}(\delta^{-1}\log^2\delta^{-1})$, then every row of the table has $2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})}$ tuples and so the entire table has at most $2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})}\log(m)$ tuples.

Once the tuples for the table $\mathcal{T}$ have been computed, in the second stage of the algorithm we consider the rows $\mathcal{T}[\ell]$ of the table where there is a tuple $(\mu'', \eta'', \text{Free}'', \text{Rsrvd}'')$ for which $\mu_i'' = 0$ for all $i \in [L]$, i.e., for which there are no constraints involving integral variables and hence the corresponding mixed integer program is in fact a linear program. We solve these linear programs using the following result.

Lemma 5.1. *We can compute a basic feasible solution for* $\text{MILP}'(\mu'', \eta'', \text{Free}'', \text{Rsrvd}'')$ *where* $\mu_i'' = 0$ *for all* $i \in [L]$ *in time* $\mathcal{O}(2^{\delta^{-1}\log^2\delta^{-1}}\log^5 n)$.

Proof. For each machine speed b_i a configuration $\gamma \in \mathcal{C}_i$ has at most δ^{-1} job sizes. By Lemma 2.1 at most $\mathcal{O}(\delta^{-1}\log\delta^{-1})$ different job sizes can appear in one such configuration, so the number of variables $x_{i,\gamma}$ in $\text{MILP}'(\mu'', \eta'', \text{Free}'', \text{Rsrvd}'')$ is $\mathcal{O}((\delta^{-1})^{\delta^{-1}\log\delta^{-1}})$ for each $i \in [\tau]$. There are $\tau = \mathcal{O}(\delta^{-1}\log(\delta^{-1}n))$ machine speeds, and thus the number of variables $x_{i,\gamma}$ is $\mathcal{O}((\delta^{-1})^{\delta^{-1}\log\delta^{-1}}\delta^{-1}\log(\delta^{-1}n)) = \mathcal{O}(2^{\delta^{-1}\log^2\delta^{-1}}\log n)$. The number of variables $y_{i,j}$ is $\mathcal{O}(\tau^2)$, so the total number of variables is $\mathcal{O}(2^{\delta^{-1}\log^2\delta^{-1}}\log^2 n)$ and the number of constraints is $\mathcal{O}(\tau)$. Using the LP solver of Lee and Sidford [18] to obtain a vertex solution (as described in Remark 6.5.2 in [9]) yields the desired result. □

5.1 Third Stage: Computing a Solution

The third stage of the algorithm starts when the row ρ of the table $\mathcal{T}$ has been reached for which in every tuple $(\mu, \eta, \text{Free}, \text{Rsrvd})$, $\mu_i = 0$ for all $i \in [L]$, so a solution for MILP′ for these tuples has been computed. For each row ℓ of the table, starting at row $\rho - 1$ and moving towards the first row, we consider each tuple $t = (\mu, \eta, \text{Free}, \text{Rsrvd})$ in row ℓ for which $\mu_i > 0$ for at least one $i \in [L]$, and each tuple $t' = (\mu', \eta', \text{Free}', \text{Rsrvd}')$ in row $\ell+1$ and try to determine whether t' was obtained from t through the application of Steps (I) to (III), and if so we extend the solution for MILP′ computed for t' into a solution for t. To do this, we first compute the changes in the values of the components of the tuples that would be needed to produce t' from t as explained in Section 4; let these changes be μ'', η'', Free'', and Rsrvd'', where $\mu'' := \mu - 2\mu'$, $\eta'' := \eta - 2\eta'$, $\text{Free}'' := 2\text{Free}' - 2 - \text{Free}$, and $\text{Rsrvd}'' := 2\text{Rsrvd}' - \text{Rsrvd}$ (the -2 term in Free'' is needed because of the $+1$ term in (4)).

If $\mu'', \eta'', \text{Free}'', \text{Rsrvd}'' \geq 0$ we try to find a solution to the following mixed integer linear program to add back the machines and jobs that would have been removed from t in Steps (I) to (III).

(MILP-DP-STEP)

$$\sum_{\gamma \in \mathcal{C}_i} x_{i,\gamma} = \mu_i'' \qquad \text{for } i \in [L']$$

$$\sum_{i=1}^{L'} \sum_{\gamma \in \mathcal{C}_i} \gamma_j x_{i,\gamma} + \sum_{i=1}^{L'} y_{i,j} = \eta''_j \qquad \text{for } j \in [L']$$

$$\sum_{i=1}^{L'} \sum_{\gamma \in \mathcal{C}_i} \gamma_j x_{i,\gamma} + \sum_{i=1}^{L'} y_{i,j} + \hat{y}_j = \eta''_j \qquad \text{for } j \in \{L'+1, \ldots, L''\}$$

$$\sum_{\gamma \in \mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{\delta b_i} \right\rceil x_{i,\gamma} + \hat{x}_i = \text{Free}''_i \qquad \text{for } i \in [L']$$

$$\sum_{i=1}^{L'} y_{i,j} \leq 1 \qquad \text{for } j \in [L']$$

$$\sum_{j=1}^{L'} \frac{b_j}{\delta b_i} y_{i,j} \leq \text{Rsrvd}''_i \qquad \text{for all } i \in [L']$$

$$\hat{x}_i \in \{0,1\}, x_{i,\gamma} \in \{0,1,2\} \qquad \text{for } i \in [L'], \gamma \in \mathcal{C}_i$$

$$y_{i,j} \in [0,1] \qquad \text{for } i \in [L'], j \in [L'']$$

$$y_j \in \{0,1\} \qquad \text{for } j \in [L'']$$

$$y_{i,j} = 0 \qquad \text{for } i \in [L'] \text{ and } b_j > \delta b_i$$

$$\hat{y}_j = 0 \qquad \text{for } i \in \{L'+1, \ldots, L'\}$$

The slack variables $\hat{x}_i$ are needed because of the rounding up in the definition of Free in (4). As explained in Step (I), a variable $x_{i,\gamma}$ could have value 2, so the constraint variables $x_{i,\gamma}$ can have values 0, 1, or 2. The variables $y_{i,j}$ and $\hat{y}_j$ account for the jobs that were removed in Steps (II) and (III), respectively. First, we show that MILP-DP-STEP has a solution.

Lemma 5.2. *If* MILP$'(\mu, \eta, \text{Free}, \text{Rsrvd})$ *and* MILP$'(\mu', \eta', \text{Free}', \text{Rsrvd}')$ *are as in Corollary 4.1.1, then MILP-DP-STEP has a solution.*

Proof. Let (x, y) be a solution to MILP$'(\mu, \eta, \text{Free}, \text{Rsrvd})$ and x'', y'' be as defined in Steps (I) to (III) and used to derive $(\mu', \eta', \text{Free}', \text{Rsrvd}')$. We show that (x'', y'') satisfies al the constraints of MILP-DP-STEP. The first two constraints and the fifth constraints are satisfied by definition. By (4), for each $i \in [L']$:

$$\text{Free}'_i = \left\lceil \frac{\text{Free}_i + \sum_{\gamma \in \mathcal{C}_i} \lceil \frac{\text{free}(i,\gamma)}{\delta b_i} \rceil x''_{i,\gamma}}{2} \right\rceil$$

Since $\text{Free}''_i = 2\text{Free}'_i - 2 - \text{Free}_i$, then $\text{Free}'' = \hat{x}_i + \sum_{\gamma \in \mathcal{C}_i} \left\lceil \frac{\text{free}(i,\gamma)}{\delta b_i} \right\rceil x''_{i,\gamma}$, where $\hat{x}_i \in \{0,1\}$. This shows that the configurations from x'' induce a solution for MILP-DP-STEP that satisfies the fourth constraints. By (5), for each $i \in [L']$:

$$begingather*]\text{Rsrvd}'_i = \left\lceil \frac{\text{Rsrvd}_i + \lceil \sum_{j=1}^{L'} \frac{b_j y''_{i,j}}{\delta b_i} \rceil}{2} \right\rceil$$

Since $\text{Rsrvd}''_i = 2\text{Rsrvd}'_i - \text{Rsrvd}_i$ then $\text{Rsrvd}''_i \geq \sum_{j=1}^{L'} \frac{b_j}{\delta b_i} y''_{i,j}$. Finally, from Steps (II) and (III),

$$\sum_{i=1}^{L} \sum_{\gamma \in \mathcal{C}_i} \gamma_j x''_{i,\gamma} + \sum_{i=1}^{L} y''_{i,j} + \hat{y}_j = \eta''_j$$

for every $j \in [L'']$. Where $\hat{y}_j = 0$ for every $j \in [L']$ and $\hat{y}_j \in \{0,1\}$ for every $j \in \{L'+1, \ldots, L''\}$ to account for the jobs removed in Step (III). □

We can compute a solution for MILP-DP-STEP efficiently using a simple dynamic programming approach.

Lemma 5.3. *MILP-DP-STEP can be solved in time* $2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})}$.

Proof. We can solve MILP-DP-STEP using a straightforward dynamic programming approach. We first consider only the integral part. The number of integral variables is $L' + L'(\delta^{-1})^{L'} = 2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})}$. Each row of the dynamic programming table stores a tuple (μ, η, F) corresponding to each one of the possible values of the right-hand sides of the first four constraints, where $\mu, F \in \mathbb{Z}_{\geq 0}^{|L'|}$ and $\eta \in \mathbb{Z}_{\geq 0}^{|L'|+\delta^{-1}\lceil\log\delta^{-1}\rceil}$. The number of possible values for μ is $\sigma^{|L'|} = 2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})}$, where σ is the support for the integer variables. The number of possible values for η is $(\delta^{-1}\sigma+1)^{|L'|+\delta^{-1}\lceil\log\delta^{-1}\rceil} = 2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})}$. The number of possible values for F is $(2\delta^{-1}\sigma)^{|L'|} = 2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})}$. Thus, the size of the table can be bounded by $2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})}$.

The first row of the dynamic programming table stores a tuple where all values are zero. For each configuration variable $x_{i,\gamma}$ we process each row of the table storing a tuple (μ, η, F) and create a tuple (μ', η', F') by adding to μ one machine of speed b_i, adding all jobs in γ to η, and adding $\sum_{\gamma \in \mathcal{C}_i} \lceil \text{free}(i,\gamma)/(\delta b_i)\rceil x_{i,\gamma}$ to F. Tuple (μ', η', F') is added to the table if not already in it. This step is performed twice for each variable $x_{i,\gamma}$. For each slack variable x_i we process each row of the table storing a tuple (μ, η, F) and add to the table tuple (μ, η, F') if the tuple is not there, where $F'_i = F_i + 1$ and F_i, F'_i are the i-th elements in F and F', respectively.

Thus, the entire table can be computed in time $2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})}$. Note that we only have to compute this table once and can use it as a lookup table afterward: We can find the solutions that our algorithmneeds by using the right-hand side of the MILP-DP-STEP as an index.

Now we only need to show how to compute the values of the variables $y_{i,j}$ and $\hat{y}_j$. Consider the values of the right-hand sides of MILP-DP-STEP: μ'', η'', Free'', 1, and Rsrvd''. We look for a tuple (μ, η, F) in the table such that $\mu = \mu''$, $F_i = \text{Free}''_i$ for all $i \in [L']$, and $\eta''_j - \eta_j \in \{0,1\}$ for all $i \in [L'']$. We set $y_j = \eta''_j - \eta_j$ for $j \in \{L'+1, \ldots, L''\}$. Let A be the set of jobs for which $\eta''_j - \eta_j = 1$ with $j \leq L'$. If the jobs in A can be scheduled in the time allocated by Rsrvd'', we can sort the jobs in non-increasing order of processing times and schedule them greedily on the corresponding machines (also sorted in non-increasing order of speed). This schedule must be valid, as otherwise there is no valid schedule for the jobs. Calculating the values for $y_{i,j}$ takes time $\mathcal{O}((n+m)\log(n+m))$ where n is the number of jobs and m is the number of machines. Thus, in our case we can compute an assignment for the fractional $y_{i,j}$ variables in time $\mathcal{O}(\delta^{-1}\log^2\delta^{-1})$ as $n = m = L' = \mathcal{O}(\delta^{-1}\log\delta^{-1})$. There are $2^{\mathcal{O}(\delta^{-1}\log\delta^{-1})}$ possibilities for the $\hat{y}_j$ variables and $2^{\mathcal{O}(\delta^{-1}\log\delta^{-1})}$ possible sets A which we have to schedule. Thus, we obtain the claimed runtime. □

Let $(x'', \hat{x}, y'', \hat{y})$ be a solution for MILP-DP-STEP, where x'' denotes the variables $x_{i,\gamma}$, y'' the variables $y_{i,j}$, $\hat{x}$ the variables $\hat{x}_i$ and $\hat{y}$ the variables $\hat{y}_j$. From this solution we can build a solution (x, y) for MILP$'(\mu, \eta, \text{Free}, \text{Rsrvd})$ from a solution (x', y') for MILP$'(\mu', \eta', \text{Free}', \text{Rsrvd}')$ as follows. Let $d \in [L]$ be such that $\mu_d > 2\mu'_d$, so a machine M_ψ of speed b_d was removed when computing $(\mu', \eta', \text{Free}', \text{Rsrvd}')$ from $(\mu, \eta, \text{Free}, \text{Rsrvd})$ using Steps (I) to (III); this machine exists because of the selection of c_h in Step (I). The jobs described by the variables $\hat{y}$ are scheduled on this machine M_ψ. Formally, this is stated in the following corollary.

Corollary 5.3.1. *A solution (x, y) for* MILP$'(\mu, \eta, \text{Free}, \text{Rsrvd})$ *is this: $x := 2x' + x''$ and $y := 2y' + y'' + y'''$, where*

$$y''' := \begin{cases} y'''_{i,j} = 1 & \text{if } i = d \text{ and } \hat{y}_j = 1 \\ y'''_{i,j} = 0 & \text{otherwise.} \end{cases}$$

Proof. Since $(x'', \hat{x}, y'', \hat{y})$ is a solution of MILP-DP-STEP, according to the definitions of μ'', η'', and MILP-DP-STEP the solution $(2x' + x'', 2y' + y'')$ schedules all jobs except for up to one job of each size b_i for $i \in \{L' + 1, \ldots, L''\}$. These extra jobs are scheduled by y''' on the machine M_ψ that was just added back. Thus, all jobs are scheduled by the solution (x, y). □

Scheduling the jobs as specified above can increase the makespan due to the definition of y''' and the error introduced by rounding Free and Rsrvd. In the next section we show that this increase can be bounded by $\mathcal{O}(\delta T)$.

5.2 The Algorithm

The algorithm builds a solution for MILP by first computing solutions for the tuples $(\mu, \eta, \text{Free}, \text{Rsrvd})$ for which $\mu_i = 0$ for all $i \in [L]$ using Lemma 5.1. Let ρ be the row of the dynamic programming table in which all tuples have this property. Then, solutions for tuples in rows $\rho - 1, \rho - 2, \ldots, 0$ are computed by solving MILP-DP-STEP and Corollary 5.3.1. The solution computed for the unique tuple in the first row of the dynamic programming table is the solution for MILP.

Lemma 5.4. *If MILP has a solution for a given makespan T then our algorithm computes a schedule of makespan $(1 + \mathcal{O}(\delta))T$.*

Proof. A solution (x, y) for the original MILP, i.e. a solution for MILP$'(\mu, \eta, 0, 0)$ where μ is the initial set of machines and η is the initial set of jobs, is obtained from solutions to a sequence of mixed integer programs:

$$\text{MILP}'\left(\mu^{(\rho)}, \eta^{(\rho)}, \text{Free}^{(\rho)}, \text{Rsrvd}^{(\rho)}\right), \ldots, \text{MILP}'\left(\mu^{(1)}, \eta^{(1)}, \text{Free}^{(1)}, \text{Rsrvd}^{(1)}\right).$$

A solution for MILP can be transformed into a schedule of makespan $(1 + \mathcal{O}(\delta))T$ by using Lemma 3.1. However, note that this schedule would not include the jobs that are removed by Steps (I) to (III). We show how to schedule these jobs. We consider first the small jobs removed in Step (I). To ensure that these jobs can be scheduled we introduced the Free variables. Every time that we apply Corollary 4.1.1 to reduce the right-hand sides of MILP$'$ we use (4) to set the value for each Free'_i, $i \in [L']$.

Because of the rounding and the factor $+1$ in equation (4), the amount of time allocated to the small jobs in Free'_i can increase by up to $\frac{3\delta b_i}{2}$ with respect to the actual amount of time that these jobs need. Hence, when considering the tuples in the ρ-th row of the dynamic programming table the total amount of extra time assigned to the variables Free'_i in that row (compared to the actual amount of time that the small removed jobs would require if this time had not been rounded up) can be computed by repeated application of (4) and so it is at most $3\delta b_i \left(\frac{1}{2} + \frac{1}{2^2} + \cdots + \frac{1}{2^\rho}\right) < 3\delta b_i$.

Let $t^{(a)} = \left(\mu^{(a)}, \eta^{(a)}, \text{Free}^{(a)}, \text{Rsrvd}^{(a)}\right)$ for any $a \in [\rho]$. Let the solution (x, y) for MILP be obtained by computing solutions for the MILP′s corresponding to tuples $t^{(\rho)}, t^{(\rho-1)}, \ldots, t^{(1)}$. In these tuples, let k_i, for each $i \in [L']$, be the biggest value such that $\mu_i^{(k_i)} > 0$, or $k_i = -1$ otherwise. Then, $\text{Free}_i^{(\ell)} = 0$ for all $\ell > k_i$. When computing a schedule for a tuple $t^{(a)}$ from a schedule for $t^{(a+1)}$ we multiply the values of the variables $\text{Free}_i^{(a+1)}$ by 2.

Hence, for computing the solution (x, y) of MILP the total amount of extra time allocated to the small jobs removed in Step (I) from machines of speed b_i, $i \in [L']$, is at most $3\delta b_i(2 + 2^2 + \cdots + 2^{k_i}) < 3\delta b_i 2^{(k_i+1)}$. Since $\mu_i^{(a+1)} \leq \frac{1}{2}\mu_i^{(a)}$ for all $a \leq k_i$ and $\mu_i^{(k_i)} \geq 1$, then the initial number of machines of speed b_i is at least 2^{k_i}. Distributing the extra amount of time needed by the variables Free_i uniformly among the machines of speed b_i and greedily scheduling the small jobs there increases the makespan on each one of these machines by at most $3\delta b_i 2^{k_i+1}/2^{k_i} = 6\delta b_i \leq 6\delta T$.

A similar argument can be used to bound the increase in makespan caused by the rounding of the variable Rsrvd in (5): By the rounding in (5) the time allocated to the jobs removed in Steps (II) and (III) in Rsrvd_i is at most δb_i larger than the actual time needed for these jobs. Hence, the total amount of extra time allocated to the variables Rsrvd_i in the ρ-th row of the dynamic programming table is at most $\delta b_i(\frac{1}{2} + \frac{1}{2^2} + \cdots + \frac{1}{2^\rho}) < \delta b_i$. Similary as above, distributing this extra time caused by rounding the variables Rsrvd among the machines of speed b_i and again greedily scheduling the small jobs there increases the makespan by at most $\delta b_i 2^{k_i+1}/2^{k_i} = 2\delta b_i \leq 2\delta T$.

To transform a tuple $t^{(a)}$ into a tuple $t^{(a+1)}$ using Steps (I) to (III) a set $\varphi^{(a)}$ of jobs of length smaller than $b_{L'}$ is removed in Step (III). Also a set $\mathcal{M}^{(a)}$ of machines of speed at least b_L is removed in Step (I). When building the solution for $\text{MILP}'\left(t^{(a)}\right)$ from the solution for $\text{MILP}'\left(t^{(a+1)}\right)$ the mixed integer program MILP-DP-STEP adds back the machines $\mathcal{M}^{(a)}$ (by setting $\mu'' = \mu - 2\mu'$) and these machines are used to schedule the jobs in $\varphi^{(a)}$. Since the processing times of the jobs in $\varphi^{(a)}$ are much smaller than the speeds of machines in $\mathcal{M}^{(a)}$, scheduling the jobs $\varphi^{(a)}$ only slightly increases the makespan, as shown below. Furthermore, the sets $\mathcal{M}^{(a)}$ are disjoint for different values of a, so the makespan increase caused by the jobs $\varphi^{(a)}$ is not cumulative.

The sizes of the jobs in $\varphi^{(a)}$ are bounded by $b_{L'}$ and all jobs in $\varphi^{(a)}$ have different processing time. Hence, the number of jobs in $\varphi^{(a)}$ is at most $L'' - L \leq \delta^{-1}\lceil \log \delta^{-1} \rceil$ by Lemma 2.1. As explained in Section 2 the job lengths are given by the equations

$$b_{k,\ell} = 2^{-k} s_{\max}\left(1 - \frac{\ell}{2\delta^{-1}}\right) \text{ for } k = 0, 1, \ldots, \lceil \log(\delta^{-1} n^2) \rceil \text{ and } \ell = 0, 1, \ldots, \delta^{-1}$$

and these values are reindexed as $b_1, b_2, \ldots, b_\tau$ so $b_i = b_{\lfloor \frac{i-1}{2} \rfloor, (i-1) \bmod \delta^{-1}}$. Therefore, the jobs

removed throughout all applications of Step (III) have total length

$$\begin{aligned}
\sum_{i=1}^{\delta^{-1}\lceil\log\delta^{-1}\rceil} b_{L'+i} &= \sum_{i=1}^{\delta^{-1}\lceil\log\delta^{-1}\rceil} 2^{-\lfloor\frac{L'+i-1}{\delta^{-1}}\rfloor} s_{\max}\left(1-\frac{(L'+i-1) \bmod \delta^{-1}}{2\delta^{-1}}\right)\\
&= \sum_{i=1}^{\delta^{-1}\lceil\log\delta^{-1}\rceil} 2^{-\lceil\log(\delta^{-3}\log\delta^{-1})\rceil-\lceil\log\delta^{-2}\rceil-\lfloor\frac{i-1}{\delta^{-1}}\rfloor} s_{\max}\left(1-\frac{(i-1) \bmod \delta^{-1}}{2\delta^{-1}}\right)\\
&= 2^{-\lceil\log(\delta^{-3}\log\delta^{-1})\rceil-\lceil\log\delta^{-2}\rceil} s_{\max} \sum_{i=0}^{\delta^{-1}\lceil\log\delta^{-1}\rceil-1} 2^{-\lfloor\frac{i}{\delta^{-1}}\rfloor}\left(1-\frac{i \bmod \delta^{-1}}{2\delta^{-1}}\right)\\
&= b_{L+1}2^{-\lceil\log\delta^{-2}\rceil} \sum_{k=0}^{\lceil\log\delta^{-1}\rceil-1}\sum_{\ell=0}^{\delta^{-1}-1} 2^{-k}\left(1-\frac{\ell}{2\delta^{-1}}\right)\\
&= b_{L+1}2^{-\lceil\log\delta^{-2}\rceil}\left(\frac{3}{4}\delta^{-1}+\frac{1}{4}\right)\sum_{k=0}^{\lceil\log\delta^{-1}\rceil-1} 2^{-k}\\
&< 2b_{L+1}\delta^2\delta^{-1} \le 2b_{L+1}\delta(1+\delta) \le 3\delta b_L \text{ for } \delta < 1/2.
\end{aligned}$$

Thus, we greedily schedule all the jobs $\varphi^{(a)}$ on one of the machines $\mathcal{M}^{(a)}$, increasing the makespan by at most $3\delta b_L \le 3\delta T$. □

Theorem 1.1. *There is an EPTAS for $Q||C_{\max}$ with runtime $2^{\mathcal{O}(\varepsilon^{-1}\log^2\varepsilon^{-1})}+\mathcal{O}(n)$.*

Proof. Our algorithm produces a schedule with makespan at most $(1+\mathcal{O}(\delta))\mathrm{OPT}$ for any constant $\delta > 0$. To compute the runtime of the algorithm first note that the preprocessing described in Section 2 can be performed in $\mathcal{O}(n+\delta^{-2}\log^2\delta^{-1})$.

The number of tuples in the dynamic programming table is $2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})}\log n$, as we can assume that the number m of machines is no larger than the number n of jobs. Therefore, the linear program of Lemma 5.1 needs to be solved at most $2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})}\log n$ times. This linear program can be solved in $\mathcal{O}(2^{\delta^{-1}\log^2\delta^{-1}}\log^5 n)$ time, and thus the total time that the algorithm spends solving the linear programs is $\mathcal{O}(2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})}\log^6 n)$.

For each row of the dynamic programming table the algorithm needs to take every tuple $(\mu,\eta,\mathrm{Free},\mathrm{Rsrvd})$ in that row and every tuple $(\mu',\eta',\mathrm{Free}',\mathrm{Rsrvd}')$ in the next row to attempt to produce a schedule corresponding to $(\mu,\eta,\mathrm{Free},\mathrm{Rsrvd})$. This requires solving MILP-DP-STEP, which can be done in time $2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})}$ by Lemma 5.3. Since there are $2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})}$ tuples per row of the table, the total time needed to process all pairs of tuples in two rows of the table is $2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})}$. The table has at most $\log n$ rows, so the time to process all pairs of tuples in the entire table is $2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})}\log n$.

The total runtime of the algorithm is $\mathcal{O}(n+2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})}\log^7 n)$ as we need $\mathcal{O}(\delta^{-1}\log(n))$ iterations of the binary search for dual approximation. If $2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})} < \log^7 n$, then the runtime is $\mathcal{O}(n)$. Otherwise, the runtime is $\mathcal{O}(n)+2^{\mathcal{O}(\delta^{-1}\log^2\delta^{-1})}$. □

References

[1] Yossi Azar and Leah Epstein. "Approximation schemes for covering and scheduling in related machines". In: *Approximation Algorithms for Combinatiorial Optimization (APPROX 1998)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. ISBN: 978-3-540-69067-2.

[2] Sebastian Berndt, Hauke Brinkop, Klaus Jansen, Matthias Mnich, and Tobias Stamm. "New Support Size Bounds for Integer Programming, Applied to Makespan Minimization on Uniformly Related Machines". In: *34th International Symposium on Algorithms and Computation (ISAAC 2023)*. Vol. 283. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. DOI: 10.4230/LIPIcs.ISAAC.2023.13.

[3] Sebastian Berndt, Max A. Deppert, Klaus Jansen, and Lars Rohwedder. "Load Balancing: The Long Road from Theory to Practice". In: *2022 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*. 2022. DOI: 10.1137/1.9781611977042.9.

[4] Lin Chen, Klaus Jansen, Wenchang Luo, and Guochuan Zhang. "An Efficient PTAS for Parallel Machine Scheduling with Capacity Constraints". In: *Combinatorial Optimization and Applications*. Cham: Springer International Publishing, 2016. ISBN: 978-3-319-48749-6.

[5] Lin Chen, Klaus Jansen, and Guochuan Zhang. "On the optimality of exact and approximation algorithms for scheduling problems". In: *Journal of Computer and System Sciences* 96 (2018). Presented at SODA 2014. DOI: 10.1016/J.JCSS.2018.03.005.

[6] Friedrich Eisenbrand and Gennady Shmonin. "Carathéodory bounds for integer cones". In: *Operations Research Letters* 34.5 (2006). DOI: https://doi.org/10.1016/j.orl.2005.09.008.

[7] M. R. Garey and David S. Johnson. ""Strong" NP-Completeness Results: Motivation, Examples, and Implications". In: *Journal of the ACM* 25.3 (1978). DOI: 10.1145/322077.322090.

[8] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G.Rinnooy Kan. "Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey". In: *Discrete Optimization II*. Vol. 5. Annals of Discrete Mathematics. Elsevier, 1979. DOI: https://doi.org/10.1016/S0167-5060(08)70356-X.

[9] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Vol. 2. Algorithms and Combinatorics. Springer, 1988. DOI: 10.1007/978-3-642-97881-4.

[10] Dorit S. Hochbaum and David B. Shmoys. "A Polynomial Approximation Scheme for Scheduling on Uniform Processors: Using the Dual Approximation Approach". In: *SIAM Journal on Computing* 17 (1988). Presented at FSTTCS 1986. DOI: 10.1007/3-540-17179-7_23.

[11] Dorit S. Hochbaum and David B. Shmoys. "Using dual approximation algorithms for scheduling problems theoretical and practical results". In: *J. ACM* 34.1 (1987). Presented at FOCS 1985. DOI: 10.1145/7531.7535.

[12] Klaus Jansen. “An EPTAS for Scheduling Jobs on Uniform Processors: Using an MILP Relaxation with a Constant Number of Integral Variables”. In: *SIAM Journal on Discrete Mathematics* 24.2 (2010). Presented at ICALP 2009. DOI: 10.1137/090749451.

[13] Klaus Jansen, Kim-Manuel Klein, Marten Maack, and Malin Rau. “Empowering the configuration-IP: new PTAS results for scheduling with setup times”. In: *Math. Program.* 195.1 (2022). DOI: 10.1007/S10107-021-01694-3.

[14] Klaus Jansen, Kim-Manuel Klein, and José Verschae. “Closing the Gap for Makespan Scheduling via Sparsification Techniques”. In: *Mathematics of Operations Research* 45.4 (2020). Presented at ICALP 2016. DOI: 10.1287/moor.2019.1036.

[15] Klaus Jansen, Alexandra Lassota, and Marten Maack. “Approximation Algorithms for Scheduling with Class Constraints”. In: *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures*. SPAA ’20. Virtual Event, USA: Association for Computing Machinery, 2020. DOI: 10.1145/3350755.3400247.

[16] Klaus Jansen and Lars Rohwedder. “On Integer Programming, Discrepancy, and Convolution”. In: *Mathematics of Operations Research* 48.3 (2023). Presented at ITCS 2019. DOI: 10.1287/moor.2022.1308.

[17] Klaus Jansen and Roberto Solis-Oba. “A Polynomial Time *OPT* + 1 Algorithm for the Cutting Stock Problem with a Constant Number of Object Lengths”. In: *Math. Oper. Res.* 36.4 (2011). DOI: 10.1287/MOOR.1110.0515.

[18] Yin Tat Lee and Aaron Sidford. “Efficient Inverse Maintenance and Faster Algorithms for Linear Programming”. In: *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*. IEEE Computer Society, 2015. DOI: 10.1109/FOCS.2015.23.