# An Efficient Polynomial Time Approximation Scheme for Minimizing the Total Weighted Completion Time on Uniformly Related Machines

**Leah Epstein**
Department of Mathematics, Faculty of Natural Sciences, University of Haifa, Israel

**Asaf Levin**
Faculty of Data and Decisions Science, The Technion, Haifa, Israel

## Abstract

We study a classic scheduling problem on *uniformly related machines* for which we show an efficient polynomial time approximation scheme (EPTAS), where an EPTAS is a fast and practical approximation scheme. For a desired approximation ratio of $1+\varepsilon$ for $\varepsilon > 0$, the running time of an EPTAS is a function of $\varepsilon$ multiplied by a polynomial function of the input length. New methods and techniques are essential in developing such improved approximation schemes, and their design is a primary goal of this research agenda. We present an EPTAS for the scheduling problem of a set of jobs on uniformly related machines so as to minimize the total weighted completion time. The problem is NP-hard in the strong sense, and therefore an EPTAS is the best possible approximation scheme for the problem, unless P=NP. Prior to our work, only a PTAS was known for the problem, while an EPTAS was known only for the special case of identical machines.

**2012 ACM Subject Classification** Theory of computation → Scheduling algorithms

**Keywords and phrases** Scheduling algorithms, Approximation schemes, Min-sum objective

**Digital Object Identifier** 10.4230/LIPIcs.WADS.2025.25

**Funding** *Asaf Levin*: Partially supported by ISF – Israel Science Foundation grant number 1467/22.

## 1 Introduction

We consider one of the most basic multiprocessor scheduling problems: scheduling on uniformly related machines with the goal of minimizing the total weighted completion time of jobs. More precisely, our scheduling problem is defined as follows. We are given a set of $n$ jobs, where each job has a positive size and a positive weight associated with it. We are also given a set of $m$ machines to be used for the processing of jobs, such that each machine has a given positive speed. Running job $j$ on machine $i$ requires the allocation of a time interval on this machine, whose length is precisely the size of $j$ divided by the speed of $i$, which is called the processing time of $j$ on $i$.

We consider non-preemptive schedules and thus every job is assigned to a machine and to a single (continuous) time slot (or interval) on that machine, under the following conditions. The length of the time slot assigned to job $j$ (on one of the machines) has to be the processing time of $j$ on that machine. A machine can process at most one job at each time so the time intervals assigned to two jobs that are assigned to a common machine do not intersect in an inner point. Given such a schedule, the completion time of job $j$ is defined as the ending point of the time interval of $j$, and the weighted completion time of $j$ is the product of its weight and its completion time. The goal is to find a schedule for which the sum of the weighted completion times of all jobs.

In the scheduling community there is a consensus that the objective of weighted sum of job completion times is the most important min-sum objective, and together with the makespan minimization these are the two most important objective functions in the scheduling literature.

A special case of our problem is that of identical machines, where each machine has speed 1. A generalization of these problems is the one where each job also has a release date associated with it, which restricts its starting time such that it cannot start running before its release date. Here, all release dates are equal to zero. Another well-motivated special case of our problem is the unweighted one, where all weights are equal to 1.

We study here a core problem in scheduling theory. We refer to [13, 27, 16, 29, 35, 5] for some recent work on the problem with respect to various aspects.

Now, we define the notions of approximation algorithms and the different types of approximation schemes. An $\mathcal{R}$-approximation algorithm for a minimization problem is a polynomial time algorithm that always finds a feasible solution of cost at most $\mathcal{R}$ times the cost of an optimal solution. The infimum value of $\mathcal{R}$ for which an algorithm is an $\mathcal{R}$-approximation is called the approximation ratio or the performance guarantee of the algorithm. A polynomial time approximation scheme (PTAS) is a family of approximation algorithms such that the family has a $(1+\varepsilon)$-approximation algorithm for any $\varepsilon > 0$. An efficient polynomial time approximation scheme (EPTAS) is a PTAS whose time complexity is of the form $f(\frac{1}{\varepsilon})$ multiplied by $poly(n)$ where $f$ is some (not necessarily polynomial) computable function, and $poly(n)$ is a polynomial function of the length of the (binary) encoding of the input. A fully polynomial time approximation scheme (FPTAS) is a stronger concept, defined like an EPTAS, but the function $f$ must be a polynomial in $\frac{1}{\varepsilon}$.

In this paper, we are interested in EPTAS's and we say that an algorithm (for some problem) has a *polynomial running time complexity* if its time complexity is of the form $f(\frac{1}{\varepsilon}) \cdot poly(n)$. Note that while a PTAS may have time complexity of the form $n^{g(\frac{1}{\varepsilon})}$, where $g$ can be polynomial or even super-exponential, this cannot be the case for an EPTAS. The notion of an EPTAS is modern and finds its roots in the FPT (fixed parameter tractable) literature (see [7, 9, 14, 31]). It was introduced in order to distinguish practical from impractical running times of PTAS's, for cases where an FPTAS does not exist (unless P=NP). In this work, we design an EPTAS for the scheduling problem defined above for which an FPTAS cannot exist unless P=NP.

The seminal work of Smith [34] established the existence of a polynomial time algorithm (of time $O(n \log n)$) for solving the problem of minimizing the total weighted completion time on a single machine. This algorithm can be described as follows. The jobs are scheduled according to a non-increasing order of their densities, starting at time zero, and without any idle time (where the density of job $j$ is the ratio between its weight and its size). This ratio is called Smith's ratio, and any tie breaking policy leads to an optimal solution. The correctness of this algorithm follows by a simple exchange argument. This algorithm generalizes the SPT (shortest-processing-time) approach for the case of equal weights. In our settings, we can conclude the following property for the problem. Once the jobs have been assigned to machines (but not to time slots), the permutation of jobs assigned to a given machine is fixed according to Smith's algorithm.

For a constant number of machines (at least two machines), the problem is NP-hard in the ordinary sense, but it is solvable in pseudo-polynomial time and has an FPTAS [32]. For the case where the number of machines is a part of the input, the problem is strongly NP-hard (see e.g. problem SS13 in [15]). The case of equal weights is polynomially solvable even for the more general case of unrelated machines [22]. The property that our problem is strongly NP-hard excludes the possibility to design an FPTAS for it, and thus an EPTAS is the best possible result (unless P=NP).

The development of good approximation algorithms for problems studied here was fairly slow. Till the late 1990's, only constant approximation algorithms were developed for min-sum scheduling problems such as the ones we study. We refer to the papers cited in

[33, 1, 8] for a survey of such results. Here we elaborate on the approximation schemes in the literature. The first approximation scheme for a special case of our problem was introduced by Skutella and Woeginger [33], which was designed for the special case of identical machines. Moreover, the last scheme is in fact an EPTAS for this special case. Shortly afterwards, Afrati et al. [1] presented a PTAS for the problem on identical machines with release dates. Their approach was generalized by Chekuri and Khanna [8], who showed the existence of a PTAS for uniformly related machines with release dates (see also [2]). The discussed schemes of [1, 8] are not EPTAS's, and they cannot be converted into EPTAS's (using known methods) even for the case without release dates studied here.

Before explaining the methods and techniques of these last schemes, as well as the limitations of those approaches, we mention the state of the art of approximation schemes for general load balancing problems on identical machines and uniformly related machines. The relation between the problem studied here and this family of problems will become clear later. In load balancing problems, the goal is to minimize functions of the vector of *machine* completion times (and not job completion times as we study here). Hochbaum and Shmoys presented the dual approximation framework and used it to show that the makespan minimization problem has a PTAS for identical machines [20] and for uniformly related machines [21]. In the dual approximation framework, the problem is converted into a bin packing type problem by fixing the machine loads. It was noted in [18] that the PTAS of [20] for identical machines can be converted into an EPTAS by using an integer program in fixed dimension instead of dynamic programming. Jansen [23] developed an EPTAS for the makespan minimization problem on uniformly related machines (see [24] for an alternative EPTAS for this problem and [25, 6] for improved time complexity EPTAS's for this problem). The $\ell_p$ norm minimization problem (of the vector of machine completion times) has an EPTAS for identical machines [3, 4], and a PTAS and an EPTAS for uniformly related machines [10, 11]. The EPTAS for the $\ell_p$ norm minimization problem on uniformly related machines presented in [11] provides a method for constructing an EPTAS for our problem if all job densities are equal. However, this does not translate into an EPTAS for the problem even if there are only a constant number of distinct job densities. After the preliminary version of our work here was posted [12], Kones and Levin [28] have used the framework we present here for the case of constant number of job densities in order to establish an alternative EPTAS for the $\ell_p$ norm minimization problem on uniformly related machines, and they used this framework to approximate a generalization of this problem.

**Previous directions.** Next, we elaborate on the known approximation schemes for related problems. Skutella and Woeginger [33] observed that for identical machines the $\ell_2$ norm minimization of the vector of machine loads is equivalent to minimizing the total weighted completion time, if the jobs have equal densities. This equivalence means that an optimal solution to one problem is an optimal solution to the other as the values of the two objective functions differ by an additive constant (which is common to all feasible solutions). Using this approach, they showed that if the ratio between the maximum density and the minimum density of jobs is upper bounded by a constant, then one can adapt the ideas of Alon et al. [3, 4] and obtain an EPTAS for this restricted setting.

The scheme [33] for the problem on identical machines follows the next ideas. First, round all the job sizes and job weights (up or down) to integer powers of $1+\varepsilon$. Next, apply randomized geometric partitioning of the jobs based on their (rounded) density, solve each sub-instance consisting of all jobs of one partition using the scheme (which is similar to the one of [3, 4]), and schedule the partial solutions for every machine sorted by non-decreasing job densities. This last step of combining the solutions for the different parts in the partition was more delicate in [33], but as noted in [1], the last step could be made much simpler. Afrati et al. [1] also noted that this approach can be extended to obtain an approximation scheme (a PTAS) for the problem without release dates in other machine environments (see the last remark in [1]). Chekuri and Khanna [8] extended the techniques of [1] to the setting of uniformly related machines (with release dates). The methods of [1, 8] fail completely when one tries to obtain an EPTAS for these problems with or without release dates (even for identical machines).

**Our results.** An EPTAS for the total weighted completion time on uniformly related machines was not known prior to our work, and it was only known for the much simpler case of identical machines. We design such an EPTAS here, using a number of methods explained in the next section. In an accompanying article (see the second part of [12]), we design an EPTAS for total weighted completion time on uniformly related machines with release dates. The current EPTAS is designed for the important special case without release dates, and while this EPTAS requires new ideas, this scheme is significantly simpler than the one of the general case. Before presenting the detailed description of the scheme and its analysis, we present an overview of the scheme. A forthcoming full version will contain additional details and omitted proofs.

## 2 An Overview of Our Scheme

In our scheme, all parameters are rounded to integer powers of a parameter $1+\delta$ (where $\delta = \frac{\varepsilon}{\Upsilon}$ for a constant $\Upsilon > 0$ that is independent of $\varepsilon$) [33, 1]. Thus, the first step is rounding the job sizes, the job weights, and the speeds of the machines to integer powers of $1+\delta$. Observe that this step has a minor effect on the performance guarantee of the approximation algorithm, and it is used to structure the input in the later steps. This is a standard approach which is suitable here. We say that a set of values is discrete if for every interval $[L, U]$ (for every $0 < L < U$) we have at most $O(\log \frac{U}{L})$ distinct values of the set in the interval. Then, we would like to consider discrete values for sizes, weights, and speeds, which gives us also discrete densities and discrete job processing times on the machines.

Next, we use the shifting technique [19] by modifying the weights of some jobs to separate the input into *bounded* instances in terms of job sets. For each bounded instance, it will hold that its jobs have sufficiently similar densities. We eliminate intervals of densities from the input (and not intervals of weights). This is done by increasing weights of jobs whose densities belong to intervals that will be forbidden, so not just one interval of densities is removed but a regular pattern of densities is removed, creating gaps between density intervals. Finding the best sequence of gaps requires running our algorithm for all options.

Once there are regular gaps between density intervals, we show that each instance can be solved separately and independently from other instances, and the solutions can be combined without further modifications. That is, we apply an approach that is similar to [33], where we replace the randomized geometric partitioning by the stronger shifting technique (resulting in a slightly simpler analysis as one of the cases in [33] of jobs of similar densities in different instances is avoided, because there will be no such cases). In our shifting procedure, we ensure the following for a pair of jobs $j$ and $j'$ such that the density of $j$ is not smaller than the density of $j'$. If $j$ and $j'$ belong to a common bounded instance, then the ratio between the density of $j$ and the density of $j'$ is upper bounded by a constant (that depends on $\varepsilon$), while if $j$ and $j'$ belong to different bounded instances the ratio between the two densities is very large (at least a larger constant that also depends on $\varepsilon$). When we obtain an approximated solution for each such bounded instance, we simply concatenate the solutions by processing each job on the machine it is assigned to (in the solution to its bounded instance) and processing the jobs assigned to each machine according to their Smith's ratios. By the separation of densities, we justify this, and conclude that the cost of the resulting solution is only slightly larger than the sum of costs of the solutions to the bounded instances. Therefore, if we are able to ensure a $(1+\delta)$-approximation for each bounded instance, we will get an EPTAS for the instance resulting from this shifting technique.

The new part, where we cannot follow previous work, is where we solve each bounded instance (due to shifting, there is a polynomial number of such instances). Unlike the approach in [33], since we deal with machines of possibly different speeds, we use configuration mixed-integer linear programs (MILPs), where we require a variable of a machine configuration to be integral if the machine is heavily loaded, that is, if it receives a large total size of jobs. A configuration is a list of jobs assigned to one machine of a certain speed, where the list of relatively small jobs is not defined precisely, but the total size is within a given short interval. In order to define the notion of a heavily loaded machine, we "guess" the (approximate) load of the fastest machine (breaking ties in favor of low indices) counting the job set of the current bounded instance. Then, for the correct guess we can upper bound the load of each one of other machines (where the load of a machine is its completion time, that is the work divided by the speed, where the work of a machine is the total size of jobs of the current bounded instance assigned to it). In this way, we obtain a valid upper bound on the work of every other machine. This upper bound on the load of every machine or on its work could be seen as an alternative to the *dual approximation method* of Hochbaum and Shmoys [20, 21] for cases where this dual approximation method cannot be applied. In that case there is one upper bound for the completion time of every machine, which is not possible here since our objective is not the makespan.

We say that a machine is heavily loaded if its assigned work is at least a constant fraction of this upper bound. By splitting the machine set into slow and fast machines, we show that a slow machine is never heavily loaded in optimal solutions. Moreover, as mentioned earlier there is a limit to how much a fast machine can be loaded (in terms of the load of the first fastest machine, which we guess approximately), and as a result, the number of heavy configurations is a constant.

A machine configuration consists of the speed of the machine having this configuration, the load of this machine, and a complete list of relatively large jobs together with their densities (the jobs are large compared to the total size of jobs assigned to this machine) where a job is specified by its size and weight but not by its identity, and additionally, a configuration contains an approximate total size of smaller jobs of each density. The partition of jobs to small and large (for a given configuration) is based on both the load and on the speed of that machine (it is not based only on the speed, in the way that it is defined in many approximation schemes). This property is essential since this problem does not allow the use of the dual approximation method. The assignment of jobs that are scheduled as small jobs is carried out by another set of assignment variables. These last assignment variables can be fractional in a feasible solution to the MILP. The job sets that are small for their configurations are different for distinct configurations due to different load and speed. Such jobs are assigned (fractionally) to configurations. Thus, for every job type (which consists of a rounded size and a rounded density), there are variables counting how many copies of the job are assigned as large jobs to each configuration (so every machine scheduled based on such a configuration receives this number of jobs of this type) and how many jobs are assigned as small jobs to such a configuration (in which case the job is not assigned to a specific copy of the configuration). We find that no jobs are treated as "sand" and every job is assigned to a configuration since (due to speeds) jobs cannot be just partitioned into small and large, but assigning jobs to specific copies of configurations will not allow us to obtain the required running time. It is required to know the speed of the machine that will receive a certain job, because the objective of the MILP has only a part of the cost incurred by small jobs, and the residue has to be added (where it depends on the speed).

As in [33], we approximate the increase of cost due to the contribution of slightly larger set of small jobs using the total size of jobs of the configuration (in the sense that the increase is minor compared to this total size), and thus, the usage of a fractional assignment becomes possible, and it can be converted in turn into an integral one by slightly increasing the cost of each configuration. As for fractional configurations, we round down the number of machines with each such configuration. The total size of jobs that remain unassigned is sufficiently small to be combined into the schedule of a heavily loaded machine (at least one such fastest machine must exist, and this is sufficient for our purposes). Using the fact that we solve a bounded instance, we conclude that slightly increasing the load of a machine with respect to its value in the optimal solution to the MILP has a minor impact on the cost of the resulting solution.

Our scheme for the bounded instance is of independent interest as it extends the methods of [23] for the $\ell_2$-norm minimization problem (the case of one density for all jobs) to obtain an EPTAS for a constant number of densities (where the constant depends on $\delta$), whereas extending the original EPTAS for this $\ell_2$-norm minimization problem [11] results in (only) a PTAS for the case of constant number of densities. Thus, presenting a new EPTAS for the $\ell_2$-norm minimization problem is indeed necessary for our generalization. We refer to Section 3 for a complete description of the approximation scheme explained in this overview, and we conclude that there is an EPTAS for our problem.

## 3 The Detailed Description and Formal Analysis of Our EPTAS

In this section we present our scheme and prove its correctness.

**Properties.** Obviously, since any job can start running at any time, an optimal solution (or schedule) never introduces idle time, and moreover, every machine runs its assigned jobs in an optimal order, that is, the jobs are sorted by Smith's ratios. As any tie-breaking policy leads to the same objective function value, we use a specific tie-breaking policy in this section. More specifically, we will always break ties in favor of running larger jobs first, and in the case of equal size jobs (of the same weight), jobs of smaller indices are scheduled earlier. We call this ordering *the natural ordering*. Thus, we can define a *solution* or a schedule as a partition of the jobs to $m$ subsets, one subset for each machine, and the order for each machine will always be the natural one. In some cases, we will compute the total weighted completion time of another permutation (not of the natural ordering). This will be done for the sake of analysis when this calculation is easier for cases where we are interested in an upper bound on the objective value and this upper bound is sufficient.

For an input $X$ and a solution $B$, we let $B(X)$ denote the output and the objective value of $B$ for the input $X$. Recall that for a job $j$, the density of $j$ is the ratio between its weight and its size. Thus, running the jobs sorted according to the natural ordering is equivalent to first sorting them by non-increasing density, and for each density, the jobs are sorted by non-increasing size, breaking ties (among equal size jobs) in favor of scheduling jobs of smaller indices earlier.

For a fixed schedule, the work of machine $i$ is the total size of jobs assigned to it, and its load is its completion time, that is, the work divided by the speed (as we only consider schedules without any idle time). Let the original input instance be denoted by $A$, where job $j$ has size $a_j > 0$ and weight $\omega_j > 0$, and machine $i$ has speed $v_i > 0$.

Let $C_j$ denote the completion time of $j$ under a given schedule, that is, the total size of jobs that run before $j$ on the same machine (including $j$), divided by the speed of this machine. We let $\Gamma_j$ be defined as $\omega_j(C_j - \frac{a_j}{2v_i})$, where $C_j - \frac{a_j}{2v_i}$ is the time when half of job $j$ is completed (known as the mean busy time of job $j$ [17]). We call these values $\Gamma$-values, and obviously, the cost of a solution is at least the sum of $\Gamma$-values. Moreover, for identical machines, the difference between the cost of a solution and the sum of the $\Gamma$-values is independent of the solution (see also [33]) whereas for uniformly related machines, this difference depends on the speeds as we establish in the following lemma.

▶ **Lemma 1.** *Consider a set of jobs $\tilde{I} \subseteq A$ assigned to machine $i$. Let $\Phi = \sum_{j\in\tilde{I}} a_j$ and let $\phi = \min_{j\in\tilde{I}} \frac{\omega_j}{a_j}$ be the minimum density of any job of $\tilde{I}$. The sum of $\Gamma$-values (and the cost) of any solution that selects to run all jobs of $\tilde{I}$ on machine $i$ (possibly with other jobs) is at least $\frac{\phi\cdot\Phi^2}{2v_i}$.*

**Rounding.** Given the original instance $A$, where job $j$ has size $a_j$ and weight $\omega_j$, and machine $i$ has speed $v_i$, we create a (rounded) instance $A'$ as follows. Let $0 < \delta \le 1/8$ be an accuracy factor, that is a function of $\varepsilon$ (where $\delta < \varepsilon$), and such that $\frac{1}{\delta}$ is an integer. The sets of jobs and machines in $A'$ are the same as in $A$. Let $s_i = (1+\delta)^{\lfloor \log_{1+\delta} v_i \rfloor}$ be the speed of machine $i$ in instance $A'$. Let $w_j = (1+\delta)^{\lceil \log_{1+\delta} \omega_j \rceil}$, and $p_j = (1+\delta)^{\lceil \log_{1+\delta} a_j \rceil}$ be the weight and size (respectively) of job $j$ in instance $A'$. That is, we round up the weights and the sizes of jobs while we round down the speeds of the machines. We will sometimes use $\Theta_j = \log_{1+\delta} \frac{w_j}{p_j}$, which is always an integer due to the rounding.

Next, we bound the increase of the approximation ratio due to this step. Our goal is to conclude that we can safely consider $A'$ instead of $A$. Let $\mathcal{O}$ and $\mathcal{O}'$ denote optimal solutions for $A$ and $A'$ respectively. Let $SOL$ denote a given solution for both instances, where a solution consists of a partition of the set of jobs to the machines. The ordering for each machine may be different for the two inputs since the natural ordering is used. Using the new notation, for a given schedule, the completion time of $j$ is still denoted by $C_j$, and its weighted completion time is $w_j \cdot C_j$.

▶ **Proposition 2.** *We have $SOL(A) \le SOL(A') \le (1+\delta)^3 \cdot SOL(A)$ and $\mathcal{O}(A) \le \mathcal{O}'(A') \le (1+\delta)^3 \cdot \mathcal{O}(A)$. Furthermore, if a solution $SOL$ satisfies $SOL(A') \le (1+k\delta)\cdot\mathcal{O}'(A')$ for some $k > 0$, then it holds that $SOL(A) \le (1+(2k+4)\delta)\cdot\mathcal{O}(A)$.*

By the last proposition, we only consider the instance $A'$, and use the obtained schedule as an output for $A$. Thus, it suffices to present an EPTAS for the rounded instance $A'$.

**On the use of $\Gamma$-values for $A'$.** Given the input $A'$, if $j$ is executed on machine $i$, then $\Gamma_j = w_j(C_j - \frac{p_j}{2s_i})$ where $C_j - \frac{p_j}{2s_i}$ is the time when half of the job is completed. We say that for a given schedule, a block of jobs is a set of jobs of equal density that are assigned to one machine to run consecutively on that machine (there may be additional jobs of the same density, each assigned to run later or earlier than these jobs). The next well-known lemma shows that the sum of $\Gamma$-values for a block of jobs is a function of their total size, their common density, and the starting time of the block, and it is independent of the other properties of these jobs (but it depends on the speed of the machine). We prove the lemma for completeness similarly to the proof of Lemma 1.

▶ **Lemma 3.** *Let $\bar{I} \subseteq A'$ be a set of jobs, all having densities equal to some $\Delta > 0$, and assume that these jobs are scheduled to run consecutively on machine $i$ starting at time $\tau$. Then $\sum_{j\in\bar{I}} \Gamma_j = \Delta \cdot \left(\tau + \frac{1}{2s_i}\sum_{j\in\bar{I}} p_j\right) \cdot \left(\sum_{j\in\bar{I}} p_j\right)$ .*

**Shifting.** Based on the value of $\delta$ we define a new parameter $\xi$ for the shifting step. Its value is a constant (as a function of $\delta$). Let $\xi = \lceil \ell \cdot \log_{1+\delta} \frac{1}{\delta} \rceil = \lceil \log_{1+\delta}(\frac{1}{\delta})^{\ell} \rceil$ for a fixed integer $3 \le \ell \le 5$. Thus, $\frac{1}{\delta^3} \le \frac{1}{\delta^\ell} \le (1+\delta)^{\xi} < \frac{1+\delta}{\delta^\ell} < \frac{2}{\delta^\ell} < \frac{1}{\delta^{\ell+1}} \le \frac{1}{\delta^6}$. Next, we use the integrality of $\frac{1}{\delta^2}$ and of $\ell$. We have $(1+\delta)^{\frac{\ell}{\delta^2}} > (1+\delta\cdot\frac{1}{\delta^2})^{\ell} > (\frac{1}{\delta})^{\ell}$, and we have $\xi \le \frac{\ell}{\delta^2} < \frac{1}{\delta^3}$. However, $(\frac{9}{8})^{\xi} \ge (1+\delta)^{\xi} \ge \frac{1}{\delta^\ell} \ge 8^3$, so $\xi \ge 53$.

We form a partition of integers based on the value of $\xi$, such that every subset has $\xi$ consecutive integers. For an integer $c \in \mathbb{Z}$, let $\Omega_c = \{c\cdot\xi+1, \dots, (c+1)\cdot\xi\}$. The next step will be to partition the collections of integers in a cyclic way. There will be $\frac{1}{\delta^{\ell+1}}$ possible collections. Let $0 \le \zeta \le \frac{1}{\delta^{\ell+1}} - 1$ be an integer. The collection for the value of $\zeta$ will consist of indices of sets for which the index modulo $\frac{1}{\delta^{\ell+1}}$ is equal to $\zeta$. We will be interested in densities that are powers of $1+\delta$ that belong to $\Omega_c$ where it holds that $(c \bmod \frac{1}{\delta^{\ell+1}}) = \zeta$, and the densities for jobs with densities that belong to such a set will be increased by a constant multiplicative factor.

We define the instance $A_\zeta$ by modifying the weights of some jobs in $A'$. For a job $j$, if for some (not necessarily positive) integer $v$, $\Theta_j \in \Omega_{v/\delta^{\ell+1}+\zeta}$ (recall that the density of $j$ is an integer power of $1+\delta$ and $\Theta_j = \log_{1+\delta}\frac{w_j}{p_j}$), then $w_j^\zeta = w_j \cdot (1+\delta)^{\xi}$, and otherwise $w_j^\zeta = w_j$. We let $\Theta_j^\zeta = \log_{1+\delta}\frac{w_j^\zeta}{p_j}$ which means that in the first case $\Theta_j^\zeta = \xi \cdot \Theta_j$ and in the second case $\Theta_j^\zeta = \Theta_j$. In the first case, $\Theta_j^\zeta \in \Omega_{v/\delta^{\ell+1}+\zeta+1}$ (where $v/\delta^{\ell+1}+\zeta+1 < (v+1)/\delta^{\ell+1}+\zeta$, so it was moved to a different density collection out of the $\frac{1}{\delta^{\ell+1}}$ collections). Furthermore, weights (and densities) are increased by a multiplicative factor of at most $(1+\delta)^{\xi} \le \frac{1+\delta}{\delta^\ell}$. For any $\zeta$, let $\mathcal{O}^\zeta$ be an optimal solution for $A_\zeta$. As the set of jobs and machines is the same in $A$, $A'$, and $A_\zeta$, the sets of feasible solutions for the three instances are the same.

**Bounding the increase of the cost due to modifying $A'$ into $A_\zeta$.** Next, we bound the increase of the cost due to the transformation from $A'$ to $A_\zeta$. As a result of increasing weights for jobs with densities in a fixed density collection, no job $j \in A_\zeta$ has a density such that $\Theta_j^\zeta = \log_{1+\delta}\frac{w_j^\zeta}{p_j} \in \Omega_{v'/\delta^{\ell+1}+\zeta}$ for any integer $v'$. Any value $(1+\delta)^{\beta}$ where $\beta \in \Omega_{v/\delta^{\ell+1}+\zeta}$ for an integer $v$ is called a forbidden density for $\zeta$, and other values $(1+\delta)^{\beta'}$ for integer values of $\beta'$ are called allowed density for $\zeta$. The next lemma compares costs for $\mathcal{O}'(A')$ (which was defined to be an optimal solution for $A'$) and $\mathcal{O}^\zeta$ for specific values of $\zeta$ (for the corresponding input $A_\zeta$). This will allow us to guess a suitable value of $\zeta$ (by enumeration) and approximate $\mathcal{O}^\zeta$ rather than $\mathcal{O}'$.

▶ **Lemma 4.** *Given a solution $SOL$, any $0 \le \zeta \le \frac{1}{\delta^{\ell+1}} - 1$ satisfies $SOL(A') \le SOL(A_\zeta)$, and there exists a value $0 \le \bar{\zeta} \le \frac{1}{\delta^{\ell+1}} - 1$ such that we have $SOL(A_{\bar{\zeta}}) \le (1+2\delta)\cdot SOL(A')$. Additionally, there exists a value $0 \le \zeta' \le \frac{1}{\delta^{\ell+1}} - 1$ such that $\mathcal{O}'(A') \le \mathcal{O}^{\zeta'}(A_{\zeta'}) \le (1+2\delta)\cdot \mathcal{O}'(A')$, and if a solution $SOL_1$ satisfies $SOL_1(A_{\zeta'}) \le (1+k'\delta)\cdot\mathcal{O}^{\zeta'}(A_{\zeta'})$ for some $k' > 0$, then $SOL_1(A') \le (1+(2k'+2)\delta)\cdot\mathcal{O}'(A')$.*

**The exhaustive enumeration algorithm implementing the shifting step.** Down below we present an algorithm that receives an input where the ratio between the maximum density of any job and the minimum density of any job is constant (as a function of $\delta$), and outputs a solution of cost at most $1+\delta$ times the cost of an optimal solution for this input. The ratio between densities will be at most $(1+\delta)^y$, where $y = \frac{\xi}{\delta^{\ell+1}} - \xi - 1$, The motivation for the value of $y$ is that every density collection has $\xi$ different densities, and since there are $\frac{1}{\delta^{\ell+1}}$ options for $\zeta$, there are $\frac{1}{\delta^{\ell+1}} - 1$ options of $\zeta$ for which the density collections were not removed. Sequences of possible densities without gaps introduced by considering $A_\zeta$ for a fixed value of $\zeta$ will contain $\xi \cdot (\frac{1}{\delta^{\ell+1}} - 1) = y + 1$ densities which are integer powers of $1+\delta$, such that the largest one will be larger than the smallest one by a multiplicative factor of at most $(1+\delta)^y$.

We next use this algorithm as a black box. For every value of $\zeta$, we apply the following process and create a schedule for the input $A_\zeta$. Afterwards, we choose a solution of minimum cost among the $\frac{1}{\delta^{\ell+1}}$ resulting solutions.

Let $0 \leq \zeta \leq \frac{1}{\delta^{\ell+1}} - 1$. Decompose the allowed densities for $\zeta$ into collections of consecutive densities that are separated by intervals of forbidden densities for $\zeta$. Since densities were rounded, two densities are consecutive if the large one is larger by a factor of exactly $1+\delta$ from the smaller one. By our construction, this results in subsets of allowed densities with very different densities, such that each subset has allowed densities in an interval of the form $[(1+\delta)^{-y}\rho, \rho]$, where $\rho = (1+\delta)^{(v/\delta^{\ell+1}+\zeta)\xi}$ for some integer $v$ (this is the last density in the density collection just before $\Omega_{v/\delta^{\ell+1}+\zeta}$), $y$ is as defined above ($y = \frac{\xi}{\delta^{\ell+1}} - \xi - 1$), and additionally there is a gap between the allowed densities of one set and another set (because there are no jobs with densities in $\Omega_{v/\delta^{\ell+1}+\zeta}$ in $A_\zeta$). More precisely, if $I$ and $I'$ are two such subsets and the allowed densities for $I$ are smaller than those of $I'$, then the largest allowed density for $I$ is smaller by a multiplicative factor of $(1+\delta)^{\xi+1} \geq \frac{1+\delta}{\delta^\ell}$ than the smallest allowed density of $I'$.

A sub-instance is defined to be a non-empty subset of the jobs corresponding to an interval of allowed densities together with the complete set of machines. Let $q$ denote the number of such (sub-)instances (where $q \leq n$). Let the instances be denoted by $I_1, \ldots, I_q$, such that the maximum allowed density in $I_p$ is $\rho_p$, and for $p > 1$, $\rho_p > \rho_{p-1}$, and in fact $\rho_p \geq (1+\delta)^{\frac{\xi}{\delta^{\ell+1}}} \cdot \rho_{p-1} \geq (\frac{1}{\delta^\ell})^{\frac{1}{\delta^{\ell+1}}} \cdot \rho_{p-1}$.

Given a set of solutions $SOL_p$, for $1 \leq p \leq q$ (where $SOL_p$ is a solution for $I_p$), define a *combined solution* $SOL$, where the jobs assigned to machine $i$ in $SOL$ are all jobs assigned to this machine in all the solutions. Obviously, in $SOL$ the jobs are scheduled sorted by non-increasing indices of their sets $I_p$.

**Bounding the cost of the combined solution.** We will prove that the effect of concatenating solutions is not harmful even though earlier solutions delay the processing of later solutions. This will hold due to the gap between densities of jobs of different instances.

▶ **Lemma 5.** *We have $OPT(A_\zeta) \geq \sum_{p=1}^{q} OPT(I_p)$. Furthermore, if for every $1 \leq p \leq q$ it holds that $SOL_p(I_p) \leq (1+\nu) \cdot OPT(I_p)$ for a fixed value $\nu > 0$, then the combined solution $SOL$ satisfies $SOL(A_\zeta) \leq (1+\nu) \cdot (1+8\delta) \cdot OPT(A_\zeta)$.*

In what follows we design a $(1+\delta)$-approximation algorithm for the bounded ratio problem, that gives a $(1+\delta)(1+8\delta)$-approximation algorithm for $A_\zeta$. Since $(1+\delta)(1+8\delta) < 1+10\delta$, for an appropriate choice of $\zeta$, we get an $(1+22\delta)$-approximation algorithm for $A'$, and a $(1+48\delta)$-approximation algorithm for $A$. Thus, letting $\delta = \frac{\varepsilon}{48}$ will give a $(1+\varepsilon)$-approximation algorithm for the general problem. The algorithm for the bounded ratio problem is applied at most $n$ times for each choice of $\zeta$, i.e., at most $\frac{n}{\delta^6}$ times in total. The reason is that we split every input $A_\zeta$ into parts with similar densities, and the number of parts cannot exceed the number of jobs.

**An EPTAS for the bounded ratio problem.** Let $I$ be a bounded instance such that all densities are in $[1, \rho = \rho(\delta) = (1+\delta)^y]$, where $y$ is a function of $\delta$ ($y$ and $\xi$ are as defined in the previous section). We will use not only the property that the number of distinct densities is a constant but also the constant maximum ratio between densities. By scaling (and possibly increasing the interval of densities), any valid input of the bounded ratio problem can be transformed into this form. We have the following properties. First, we have $y = \frac{\xi}{\delta^{\ell+1}} - \xi - 1 \geq \frac{\xi}{\delta^{\ell}} > 400$ (since $\frac{1}{\delta^{\ell+1}} - \frac{1}{\delta^{\ell}} - 1 = \frac{1-\delta-\delta^{\ell+1}}{\delta^{\ell+1}} > 1 > \frac{1}{\xi}$ as $2\delta^{\ell+1} + \delta < 1$ by $\ell \geq 3$ and $\delta \leq \frac{1}{8}$, and since $\xi \geq 53$). Second, $y + 2 < y + \xi + 1 \leq \frac{\xi}{\delta^6} \leq \frac{1}{\delta^9}$ since $\ell \leq 5$ and $\xi \leq \frac{1}{\delta^3}$ and due to integrality, $y + 3 \leq \frac{1}{\delta^9}$. Third, we also have (using the properties of $\xi$) that $(1+\delta)^y \geq (1+\delta)^{\frac{\xi}{\delta^{\ell}}} \geq \left(\frac{1}{\delta^{\ell}}\right)^{\frac{1}{\delta^{\ell}}} \geq (\frac{1}{\delta^3})^{\frac{1}{\delta^3}} > \frac{1}{\delta^{1500}}$, and $(1+\delta)^y \leq (1+\delta)^{\xi/\delta^{\ell+1}} \leq (\frac{1}{\delta^{\ell+1}})^{\frac{1}{\delta^{\ell+1}}} \leq (\frac{1}{\delta})^{6/\delta^6} \leq (\frac{1}{\delta})^{1/\delta^7 - 12}$, and last, we conclude that $y + 1 \leq \delta^2(1+\delta)^y$ since $y + 1 \leq \frac{1}{\delta^9}$ while $(1+\delta)^y \geq \frac{1}{\delta^{1500}}$.

Since the density of every job is in $[1, (1+\delta)^y]$, every job $j \in I$ has $p_j = (1+\delta)^k$, $w_j = (1+\delta)^{k'}$, for some integers $k, k'$ such that $0 \leq k' - k \leq y$. Let $\gamma = \frac{\delta^{12}}{(1+\delta)^y}$. We have $\gamma \geq \delta^{1/\delta^7}$ and $\gamma \leq \delta^{1512}$. Let $\mathcal{I}$ denote the set of values $i$ such that there is at least one job of size $(1+\delta)^i$. Clearly, $|\mathcal{I}| \leq n$. Let $n_{r,i}$ denote the number of jobs of size $(1+\delta)^i$ and density $(1+\delta)^r$. By scaling the machine speeds (that are integer powers of $1+\delta$) and possibly reordering the machines, let the speeds of machines be $s_1 \geq s_2 \cdots \geq s_m$, where without loss of generality, $s_1 = 1$, and for $i \geq 2$, $s_i = (1+\delta)^{k_i}$, for some non-positive integer $k_i \leq 0$. It is possible that there is more than one machine of speed 1, and we consider all such machines in the next paragraph.

**First guessing step and its analysis.** Recall that the work of a machine is the total size of jobs assigned to it. We would like to assume that job sizes are scaled such that the work of the most loaded machine (the most loaded machine in terms of work without taking speeds into account) of speed 1 is in $[1/(1+\delta), 1)$ (while since we have jobs of varying densities, the work of slower machines may be arbitrary). We will now show that this is possible. For a job $j$, and $1 \leq b \leq \frac{n}{\delta}$, let $D_{j,b}$ be the interval $[(1+\delta)^{b-1}p_j, (1+\delta)^b p_j)$ (we have that $(1+\delta)^{\frac{n}{\delta}} \geq n+1$ holds by a direct calculation since $\frac{n}{\delta}$ is integral).

Without loss of generality we consider solutions where at least one machine of speed 1 has at least one job. This holds since otherwise it is possible to move an arbitrary job to such a machine and get a solution of a smaller cost.

▶ **Lemma 6.** *For any solution, there exist a job $1 \leq j' \leq n$ and an integer $1 \leq b' \leq \frac{n}{\delta}$, such that the work of the most loaded machine out of works of machines of speed 1 is in $D_{j,b'}$.*

For every choice of a pair $j, b$ ($\frac{n^2}{\delta}$ choices in total), we scale (i.e., divide) job sizes by $(1+\delta)^b p_j$ (which is an integer power of $1+\delta$), and apply the algorithm described later in this section (this algorithm enforces the existence of a fastest machine with a suitable load, where it is possible that there is no such solution for some choices). By the claim, given an optimal solution $OPT$, for at least one choice of $j, b$ the assumption regarding the work of the most loaded machine of speed 1 (that the work is at least $(1+\delta)^{-1}$ and below 1) holds as a result of the scaling. We will pick the best solution, whose objective function value cannot be larger than that of the solution obtained for the correct choice of $j, b$. In what follows we analyze the properties of the correct choice of $j, b$ for a given optimal solution $OPT$ after the scaling. Moreover, the job sizes are according to the scaling.

**Machine configurations – preliminaries.** Let $U$ be a threshold on job sizes separating jobs seen as relatively large and jobs seen as relatively small. Let $\hat{I}$ be the subset of jobs assigned to machine $i$ in some solution. The $U$-cost of machine $i$ in this solution is defined as the total weighted completion time of jobs whose sizes are at least $U$, plus the $\Gamma$-values of jobs whose sizes are below $U$. The cost for machine $i$ is therefore its $U$-cost plus $\frac{1}{2s_i}\sum_{j\in\hat{I},p_j<U} w_j \cdot p_j$. Obviously, the $U$-cost never exceeds the cost, for any value of $U$. Additionally, similarly to the cost, the $U$-cost for a fixed value of $U$ is monotone in the sense that removing a job from the machine decreases its $U$-cost, and thus, if we consider a block of jobs of a common density each of which shorter than $U$, decreasing the total size of jobs in a block also decreases the $U$-cost.

We use the following two functions of $\delta$: $f(\delta)$ and $g(\delta)$, both integral, non-decreasing, and *negative*. A machine $i$ is called *slow* if its speed is at most $(1+\delta)^{f(\delta)}$ (and otherwise we say that it is not slow or that it is fast). We say that a machine is lightly loaded if its work does not exceed $(1+\delta)^{g(\delta)}$ (and otherwise it is heavily loaded). Let

$$f(\delta) = g(\delta) - \frac{2}{\delta^2} \text{ and } g(\delta) = -(\frac{1}{\delta})^{\frac{1}{\delta^{300}}}.$$

Consider a fixed optimal solution $OPT$ (for the input considered in this section which consists of a subset of jobs, and given the scaling of machine speeds and job sizes).

▶ **Lemma 7.** *No machine has a load strictly above $\frac{2}{\delta}$ in $OPT$, and this is an upper bound on the work of any machine as well. Furthermore, in $OPT$ every slow machine is lightly loaded.*

**Formal definition of machine configuration.** Next, we define machine configurations. A configuration is a vector that defines most properties for the schedule of one machine. This consists of a set of jobs assigned to it in terms of the types of jobs, that is, a job is specified by its size and weight but not by its identity, and jobs that are relatively small are defined by their approximate total sizes and densities. The set of all configurations will be denoted by $\mathcal{C}$. For a configuration $C \in \mathcal{C}$, the first component is an integer $j_1(C) \in \mathbb{Z}$, such that the total work of the machine is in $((1+\delta)^{j_1(C)-2}, (1+\delta)^{j_1(C)}]$. The second component is a non-positive integer $j_2(C)$ such that the speed of the machine is $(1+\delta)^{j_2(C)}$. The third component is an integer $j_3(C) \in \mathbb{Z}$, such that $j_3(C) = j_1(C) - j_2(C)$, and therefore the completion time (or load) of the machine is in $((1+\delta)^{j_3(C)-2}, (1+\delta)^{j_3(C)}]$.

Recall that $\gamma = \frac{\delta^{12}}{(1+\delta)^y}$. For integers $r, i$ such that $0 \le r \le y$, $i \le j_1(C)$, and $\gamma(1+\delta)^{j_1(C)-1} \le (1+\delta)^i \le (1+\delta)^{j_1(C)}$ (i.e., $j_1(C) - 1 + \lceil \log_{1+\delta}\gamma \rceil \le i \le j_1(C)$), there is an integer component $n_{r,i}(C) \ge 0$ stating how many jobs of size $(1+\delta)^i$ and density $(1+\delta)^r$ are assigned to this machine. These jobs are called *large* (large for configuration $C$, or large for a machine that has configuration $C$). We let $n_{r,i}(C) = 0$ for other (smaller or larger) values of $i$ (these are constants that are not a part of the configuration). There are additional components for other jobs assigned to a machine whose configuration is $C$. These jobs (which are not taken into account in the components of the form $n_{r,i}(C)$) must have smaller values of $i$, as jobs with larger values of $i$ (jobs of sizes above $(1+\delta)^{j_1(C)}$) cannot be assigned to a machine that has configuration $C$. This is so since they are too large, given the upper bound on the work of the machine. These remaining jobs are called *small* jobs for $C$, or small jobs for a machine whose schedule is according to configuration $C$. For every $r$, there is an integer component $t_r(C) \ge 0$ such that the total size of small jobs (of sizes in $(0, \gamma(1+\delta)^{j_1(C)-1})$) of density $(1+\delta)^r$ assigned to a machine with configuration $C$ is in $((t_r(C)-1)\cdot\gamma(1+\delta)^{j_1(C)-1}, t_r(C)\cdot\gamma(1+\delta)^{j_1(C)-1}]$. We have $t_r(C) = 0$ if and only if there are no such jobs for a machine with this configuration.

**Valid configurations.** Recall that $\mathcal{I}$ is the set of indices $i$ such that there is at least one job of size $(1+\delta)^i$. A configuration $C$ is valid if the total size of jobs is sufficiently close to its required work, and its load does not exceed $\frac{2}{\delta}$. Formally, letting $\mathcal{I}(C) = \{i \in \mathcal{I} : j_1(C) - 1 + \lceil \log_{1+\delta} \gamma \rceil \le i \le j_1(C)\}$ it is required that $\sum_{0 \le r \le y, i \in \mathcal{I}(C)} (1+\delta)^i \cdot n_{r,i}(C) + \sum_{r=0}^{y} t_r(C) \cdot \gamma(1+\delta)^{j_1(C)-1} \in \big((1+\delta)^{j_1(C)-2}, (1+\delta)^{j_1(C)+1}\big]$. The reason for the increased exponent of the right endpoint is that for every density the total size of small jobs may be smaller by an additive term of $\gamma(1+\delta)^{j_1(C)-1}$ compared to its upper bound, and since $(y+1)\cdot\gamma < \delta$, it suffices to increase the right endpoint by a multiplicative factor of $1+\delta$. We also let $\mathcal{I}'(C) = \{i \in \mathcal{I} : i \le j_1(C) - 2 + \lceil \log_{1+\delta} \gamma \rceil\}$, and require $(1+\delta)^{j_3(C)-2} < \frac{2}{\delta}$. In the remainder of this section, given $C$, we will use the threshold $U(C) = \gamma(1+\delta)^{j_1(C)-1}$ for computing the $U$-cost of a machine with configuration $C$ (this is the value $U$ of the configuration $C$).

**Bounding the number of configurations.** The number of components is therefore at most $3+(y+1)\cdot(\lceil \log_{1+\delta} \frac{1}{\gamma} \rceil + 2) + (y+1) \le 3 + (y+1)(\log_{1+\delta} \frac{1}{\gamma} + 3)$. Since $\log_{1+\delta} \frac{1}{\gamma} = y + \log_{1+\delta} \frac{1}{\delta^{12}} \le y + \frac{1}{\delta^{13}}$, and $y+1 \le \frac{1}{\delta^9}$, we have $3 + (y+1)(\log_{1+\delta} \frac{1}{\gamma} + 3) \le 3 + \frac{1}{\delta^9}(\frac{1}{\delta^9} + \frac{1}{\delta^{13}} + 3) < \frac{1}{\delta^{23}}$. We will use the following bounds.

▶ **Lemma 8.** *For every pair of value $j_1$ and $j_2$, there are at most $(\frac{1}{\delta})^{1/\delta^{33}}$ configurations $C$ with $j_1(C) = j_1$ and $j_2(C) = j_2$.*

▶ **Lemma 9.** *The number of options for $j_1, j_2, j_3$ (for which there are valid configurations) is $O(\frac{n^2 m}{\delta})$*

Let $J = \{\sigma_1, \sigma_2, \dots, \sigma_\kappa\}$ be the set of all $\kappa$ different speeds of machines (where $1 \le \kappa \le m$) such that $1 = \sigma_1 > \sigma_2 > \cdots > \sigma_\kappa$, and let $N_i \ge 1$ be the number of machines of speed $\sigma_i$.

**An auxiliary mathematical program.** Consider the following mathematical program $\mathcal{P}$. The goal of $\mathcal{P}$ is to determine a partial schedule via machine configurations. For every configuration $C$, $X_C$ is a decision variable stating how many machines have this configuration. Obviously, the number of used configurations whose second component is $\sigma_i$ cannot exceed $N_i$ (it is not required to use all machines, but it is required to assign all jobs). For every triple of density, size, and a configuration (density $(1+\delta)^r$, size $(1+\delta)^i$, and configuration $C$), there is a decision variable $Y_{r,i,C}$ corresponding to the number of jobs of this size and density that are assigned to machines whose configurations are $C$ as small jobs for this configuration. The variable $Y_{r,i,C}$ may be positive only if a job of size $(1+\delta)^i$ is small for configuration $C$, i.e., $i \le j_1(C) - 2 + \lceil \log_{1+\delta} \gamma \rceil$, and in all other cases we set $Y_{r,i,C} = 0$ to be a constant rather than a variable. Each such variable is used for all copies of $C$ together since jobs assigned as small jobs within the copies of the configuration may be of different sizes. Based on these decision variables we define a set of constraints and solve the resulting mixed-integer linear program $\mathcal{P}$.

A configuration $C$ has a cost denoted by $cost(C)$ associated with it, which is the $U$-cost for the threshold $U(C) = \gamma(1+\delta)^{j_1(C)-1}$. Recall that for the purpose of calculating the $U$-cost of a machine, a list of its large jobs is needed. However, for its jobs of size below $U$ (i.e., small jobs), the exact list of such small jobs is not needed. The only needed property of the subset of small jobs (for configuration $C$) of each one of the densities is their total size. This allows us to assign very different jobs that are small to machines with the same configuration, as long as these jobs are indeed small for the configuration and have the same density.

For large jobs, the exact identities of jobs are not needed as well, but a list of densities and sizes is needed. Thus, the $U$-cost for configuration $C$ is calculated assuming that the machine has exactly $n_{r,i}(C)$ jobs of size $(1+\delta)^i$ and weight $(1+\delta)^{i+r}$ (i.e., density $(1+\delta)^r$) assigned to it, and the total size of small jobs (of sizes in $(0, \gamma(1+\delta)^{j_1(C)-1})$) with densities equal to $(1+\delta)^r$ is exactly $t_r(C) \cdot \gamma(1+\delta)^{j_1(C)-1}$. The objective of $\mathcal{P}$ is to minimize the sum of costs of configurations plus the missing parts of the costs of jobs that are assigned as small. For that, for a job of size $(1+\delta)^i$ and density $(1+\delta)^r$ that is assigned to a machine of speed $s$ as a small job for its configuration, an additive term of $\frac{(1+\delta)^{r+2i}}{2s}$ is incurred (this is the difference between the actual cost for this job, and its $\Gamma$-value, which is the part of the cost already included in the $U$-cost). This last term only depends on the speed of the machine that runs the job rather than the specific machine. Thus, for each $r$ and $i$ we add

$$(1+\delta)^{r+2i} \cdot \sum_{C \in \mathcal{C}} \frac{Y_{r,i,C}}{2(1+\delta)^{j_2(C)}}$$

to the cost of configurations to get the total cost of the schedule, where $\sum_{C \in \mathcal{C}} Y_{r,i,C}$ is the number of jobs of size $(1+\delta)^i$ and density $(1+\delta)^r$ that are assigned as small jobs for their configurations.

$$\min \sum_{C \in \mathcal{C}} cost(C) \cdot X_C + \sum_{r=0}^{y} \sum_{i \in \mathcal{I}} (1+\delta)^{r+2i} \cdot \sum_{C \in \mathcal{C}} \frac{Y_{r,i,C}}{2(1+\delta)^{j_2(C)}} \qquad s.t.$$

$$\sum_{C \in \mathcal{C}: (1+\delta)^{j_2(C)} = \sigma_i} X_C \le N_i \qquad \forall \sigma_i \in J \qquad (1)$$

$$\sum_{C \in \mathcal{C}} n_{r,i}(C) \cdot X_C + \sum_{C \in \mathcal{C}} Y_{r,i,C} = n_{r,i} \qquad \forall 0 \le r \le y, i \in \mathcal{I} \qquad (2)$$

$$\sum_{i \in \mathcal{I}'(C)} (1+\delta)^i \cdot Y_{r,i,C} - \left( (t_r(C)+1) \cdot \gamma \cdot (1+\delta)^{j_1(C)-1} \right) \cdot X_C \le 0 \qquad (3)$$

$$\forall 0 \le r \le y, C \in \mathcal{C}$$

$$\sum_{C \in \mathcal{C}: j_1(C) \in \{-1,0,1\}, j_2(C)=0} X_C \ge 1 \qquad (4)$$

$$Y_{r,i,C} \ge 0 \qquad \forall C \in \mathcal{C}, 0 \le r \le y, i \in \mathcal{I}'(C) \qquad (5)$$

$$X_C \ge 0 \qquad \forall C \in \mathcal{C} \qquad (6)$$

**Motivation for the constraints.** Later, we will use the last mathematical program to direct our algorithm. Before we continue we explain the intuition behind its constraints. We stress that the formal proof does not use this intuition, so it is given to assist the reader. Condition (1) ensures that the number of used machines for each speed does not exceed the existing number of such machines. Condition (2) states that every job is assigned (either it is a large job or a small job). Condition (3) considers jobs of density $(1+\delta)^r$ that are assigned as small to machines scheduled according to configuration $C$, and verifies that sufficient space is allocated for them if the space for them is slightly extended (this extended space may be used as we show later). Condition (4) ensures that indeed there is a machine of the maximum speed that has a work that is close to 1, that is, above $\frac{1}{(1+\delta)^2}$ and at most $1+\delta$ (the condition that the work is in $[1/(1+\delta), 1)$ is slightly relaxed).

Observe that we distinguish between large and small jobs for a machine based on the configuration and not solely by its speed. This feature is needed since our problem does not allow the use of the dual-approximation method (because the objective is not based on machine loads).

**Formal definition of the subset of integer variables.** We define the set of heavy configurations $\mathcal{C}_H$ as $\mathcal{C}_H = \{C \in \mathcal{C} : j_1(C) > g(\delta)\}$, (note that the definition of a heavy configuration or a heavily loaded machine depends only on the work, and it is independent of the speed) and the complement set of light configurations is $\mathcal{C}_L = \mathcal{C} \setminus \mathcal{C}_H$. By Lemma 7 it is sufficient to consider light configurations for slow machines. Therefore, we delete heavy configurations for slow machines and do not consider them (which is justified later in the comparison between the cost of an optimal schedule and the optimal solution of $\mathcal{P}$).

We see $\mathcal{P}$ as a mixed-integer linear program and thus we define next the set of variables that are forced to be integral and the set of variables that are allowed to be fractional. All *variables* $Y_{r,i,C}$ may be fractional. The variables of (light) configurations corresponding to slow machines and variables of light configurations of fast machines may be fractional, whereas the variables of heavy configurations corresponding to fast machines must be integral. The number of variables is polynomial, and the number of integral variables will be constant (a function of $\delta$).

Recall that for every pair of speed and work (i.e, a pair $j_1, j_2$), the number of different valid configurations is constant (as a function of $\delta$). The number of different speeds of fast machines is at most $|f(\delta)| = -f(\delta)$. The number of different values of $j_1(C)$ such that $C$ is a heavy configuration is at most $-g(\delta) + \lceil \log_{1+\delta} \frac{2}{\delta} \rceil \leq \frac{2}{\delta^2} - g(\delta) = -f(\delta)$, and $j_2(C)$ of a heavy configuration $C$ must be a fast speed (so there are at most $-f(\delta)$ values for it). As the number of integral variables is constant (as a function of $\delta$), an optimal solution can be found in polynomial time [30, 26] that is a function of $\delta$ multiplied by a polynomial in the input encoding length.

**Presenting a feasible solution to $\mathcal{P}$ whose cost is at most the cost of an optimal schedule.** We will first compare the cost of the optimal schedule $OPT$ to the cost of an optimal solution of $\mathcal{P}$. Later, we will show how to obtain an actual schedule given a solution of $\mathcal{P}$, such that the cost of the schedule is larger only by a factor of at most $1+\delta$ than the objective function value of the solution to $\mathcal{P}$. Let $(X^*, Y^*)$ denote an optimal solution to the mixed-integer linear program, and let $Z^*$ be its objective value. Let $Z^{OPT}$ denote the cost of $OPT$.

▶ **Theorem 10.** *If $(X^*, Y^*)$ is an optimal solution of $\mathcal{P}$, then $Z^* \leq Z^{OPT}$.*

**Using the solution $(X^*, Y^*)$ for finding an approximate schedule.** Our last rounding steps are designed in order to use $(X^*, Y^*)$ for getting an approximate schedule for the rounded ratio problem. These steps are the constructive methods for establishing our scheme.

We start with constructing an alternative set of values of the variables and bounding the resulting cost from above. We will later convert this alternative solution (that may violate some of the constraints of $\mathcal{P}$) into a schedule. For every $C \in \mathcal{C}$, let $X'_C = \lfloor X^*_C \rfloor$. If $X^*_C = 0$, then we set $X'_C = 0$. In this case $Y^*_{r,i,C} = 0$ must hold for all $r, i$ by (3), and we set $Y'_{r,i,C} = 0$. If $X^*_C > 0$, we consider two cases. If $C \in C_H$, then the variables of configurations are integral, and therefore $X'_C = X^*_C$. In this case we set $Y'_{r,i,C} = \lceil Y^*_{r,i,C} \rceil$ for any $0 \leq r \leq y$, $i \in \mathcal{I}'(C)$. Otherwise, that is, in the case $C \in C_L$, let $Y'_{r,i,C} = \left\lfloor \frac{X'_C}{X^*_C} \cdot Y^*_{r,i,C} \right\rfloor$.

Using these definitions, in the case $Y'_{r,i,C} > 0$, if $C \in C_H$, then we have $Y'_{r,i,C} \le Y^*_{r,i,C} + 1$, and otherwise we have $Y'_{r,i,C} \le \frac{X'_C}{X^*_C} \cdot Y^*_{r,i,C}$. Moreover, if $C \in C_L$ and $X^*_C > 0$, then we have $Y'_{r,i,C} > \frac{X'_C}{X^*_C} \cdot Y^*_{r,i,C} - 1 > Y^*_{r,i,c} - \frac{Y^*_{r,i,C}}{X^*_C} - 1$, by $X^*_C < X'_C + 1$ (that holds even if $X'_C = 0$).

**Bounding the cost of $(X', Y')$.** We will define a schedule and find an upper bound on its objective function value. Before that, we analyze the first increase in the cost which is due to the transformation from $(X^*, Y^*)$ to $(X', Y')$, and show that it is by an additive factor of at most $\delta^{100} \cdot Z^*$. The solution $(X', Y')$ that we consider is not a feasible solution for $\mathcal{P}$, as (for example) constraint (2) does not necessarily hold as it is possible that $\sum_{C \in \mathcal{C}} n_{r,i}(C) X'_C + \sum_{C \in \mathcal{C}} \cdot Y'_{r,i,C} \neq n_{r,i}$, but we can still bound its objective function value using the objective function value of the optimal (and feasible) solution.

Since $X'_C \le X^*_C$ for $C \in \mathcal{C}$, and $Y'_{r,i,C} \le Y^*_{r,i,C}$ for all $C \in C_L$, $0 \le r \le y$, and $i \in \mathcal{I}'(C)$, the objective function value for these variables is at most $Z^*$ plus

$$\sum_{r=0}^{y} \sum_{i \in \mathcal{I}} (1+\delta)^{r+2i} \sum_{C \in C_H : i \in \mathcal{I}'(C)} \frac{Y'_{r,i,C} - Y^*_{r,i,C}}{2(1+\delta)^{j_2(C)}}$$
$$\le \sum_{r=0}^{y} \sum_{C \in C_H} \sum_{i \in \mathcal{I}'(C)} (1+\delta)^{r+2i} \frac{X^*_C}{2(1+\delta)^{j_2(C)}} \ ,$$

since for $C \in C_H$ it holds that $Y'_{r,i,C} = 0$ if $X^*_C = 0$, and otherwise $Y'_{r,i,C} - Y^*_{r,i,C} \le 1 \le X^*_C$. Next, we show that

$$\sum_{r=0}^{y} \sum_{i \in \mathcal{I}'(C)} (1+\delta)^{r+2i} \frac{1}{2(1+\delta)^{j_2(C)}} \le \delta^{100} \cdot cost(C)$$

for any $C \in C_H$, and thus we will conclude that the objective function value of the set of variables $X'_C$, $Y'_{r,i,C}$ is at most $(1+\delta^{100}) \cdot Z^*$. We have $cost(C) \ge \frac{(1+\delta)^{2j_1(C)-4}}{2(1+\delta)^{j_2(C)}}$ (by Lemma 1, since $cost(C)$ is computed for jobs of total size above $(1+\delta)^{j_1(C)-2}$, and the jobs densities are no smaller than 1). Let $i' = \max \mathcal{I}'(C)$. We have

$$(1+\delta)^{2i'} \le (\gamma(1+\delta)^{j_1(C)-1})^2$$

and

$$\sum_{i \le i'} (1+\delta)^{2i} < (1+\delta)^{2i'+1}/\delta \ .$$

Additionally, $\sum_{r=0}^{y} (1+\delta)^r < (1+\delta)^{y+1}/\delta$. For a given $C \in C_H$, we get

$$\sum_{r=0}^{y} \sum_{i \in \mathcal{I}'(C)} (1+\delta)^{r+2i} \frac{1}{2(1+\delta)^{j_2(C)}} \le \frac{(1+\delta)^{y+2i'+2}}{2\delta^2(1+\delta)^{j_2(C)}} \le \frac{(1+\delta)^{y+2}(\gamma(1+\delta)^{j_1(C)-1})^2}{2\delta^2(1+\delta)^{j_2(C)}}$$
$$= \frac{(1+\delta)^{y+2} \frac{\delta^{24}}{(1+\delta)^{2y}} (1+\delta)^{2j_1(C)-2}}{2\delta^2(1+\delta)^{j_2(C)}} \le cost(C) \cdot \delta^{22}(1+\delta)^{4-y} \le \delta^{100} \cdot cost(C)$$

(since $(1+\delta)^y \ge \frac{1}{\delta^{1500}}$). Thus the claim regarding the cost of $(X', Y')$ holds.

**Defining a schedule of most jobs while leaving a small subset of unassigned jobs.** We will define a schedule that is based on the rounded variables. The schedule will require additional space for jobs that are small for their configurations (in some configurations) both due to rounding of variables, and the actual assignment of jobs to machines which does not exist in $\mathcal{P}$ (for such jobs). Due to the rounding of numbers of copies of configurations, some jobs that are large for their configurations will also remain unassigned. Such jobs will obviously have to be assigned, thus increasing the cost. For the resulting schedule, the objective function value of $\mathcal{P}$ may be no longer correct, though we show that the true objective is not much larger.

Next, we find additional upper bounds that are based on the rounded variables (only those that were rounded up). We prove upper bounds on the space used for jobs that are small for their configurations where a possible increase is due to the rounding of variables only. Additional space to be used for such jobs will be defined and analyzed later, and here we only analyze the space that was already defined.

Using condition (3), for every $C \in C_L$ such that $X^*_C > 0$ and $0 \le r \le y$, we have

$$\sum_{i \in \mathcal{I}'(C)} (1+\delta)^i \cdot Y'_{r,i,C} \le \sum_{i \in \mathcal{I}'(C)} (1+\delta)^i \cdot \frac{X'_C}{X^*_C} \cdot Y^*_{r,i,C}$$

$$\le \frac{X'_C}{X^*_C} \cdot (t_r(C)+1) \cdot \gamma \cdot (1+\delta)^{j_1(C)-1} \cdot X^*_C = (t_r(C)+1) \cdot \gamma \cdot (1+\delta)^{j_1(C)-1} \cdot X'_C \ .$$

Similarly, for every $C \in C_H$ such that $X^*_C > 0$ and $0 \le r \le y$, we have

$$\begin{aligned} &\textstyle\sum_{i \in \mathcal{I}'(C)} (1+\delta)^i \cdot Y'_{r,i,C} \le \sum_{i \in \mathcal{I}'(C)} (1+\delta)^i \cdot (Y^*_{r,i,C}+1) \\ &\le (t_r(C)+1) \cdot \gamma \cdot (1+\delta)^{j_1(C)-1} \cdot X^*_C + \textstyle\sum_{i \in \mathcal{I}'(C)} (1+\delta)^i \\ &\qquad \le (t_r(C)+2+\tfrac{2}{\delta}) \cdot \gamma \cdot (1+\delta)^{j_1(C)-1} \cdot X'_C \end{aligned}$$

as $X'_C = X^*_C \ge 1$, and

$$\sum_{i \in \mathcal{I}'(C)} (1+\delta)^i \le \gamma(1+\delta)^{j_1(C)-1} \sum_{i=0}^{\infty} \frac{1}{(1+\delta)^i} = \gamma(1+\delta)^{j_1(C)-1} \cdot \frac{1+\delta}{\delta} \ .$$

If $X^*_C = 0$, then the corresponding upper bound (the first one if $C \in C_L$, and the second one if $C \in C_H$) hold trivially, as all variables are equal to zero.

We let $n'_{r,i} = n_{r,i} - \sum_{C \in \mathcal{C}} n_{r,i}(C) X'_C - \sum_{C \in \mathcal{C}} Y'_{r,i,C}$. If $n'_{r,i} > 0$, then we say that $n'_{r,i}$ is the number of unassigned jobs of size $(1+\delta)^i$ and density $(1+\delta)^r$. These jobs will be called *unassigned jobs*, as they will remain unassigned if we assign jobs exactly using configurations that are based on the modified set of variables (including the assignment to configurations of jobs that are small for their configurations). We explain this assignment, which is called the first assignment step, before we proceed.

The first assignment step will be to assign jobs according to the configurations for which $X'_C > 0$, such that there will be $X'_C$ machines whose sets of large jobs will be as required (based on the definition of $C$). We will then assign $Y'_{r,i,C}$ jobs of size $(1+\delta)^i$ and density $(1+\delta)^r$ to the machines whose configuration is $C$ (these are small jobs for $C$). In order to ensure that all jobs can be scheduled, for each machine that is assigned the configuration $C$, additionally to total size of at most $t_r(C) \cdot \gamma \cdot (1+\delta)^{j_1(C)-1}$ jobs of density $(1+\delta)^r$ that are small jobs for configuration $C$ that the machine can contain, such jobs of total size at most $\frac{3\gamma}{\delta} \cdot (1+\delta)^{j_1(C)-1}$ are allocated to this machine. These jobs are called *additional jobs*, and we will calculate the increase in the total cost as a result. The unassigned jobs will all be assigned to a machine of speed 1 whose configuration $C'$ has $j_1(C') = 0$ or $j_1(C') = -1$. We will compute an upper bound on the total size of these unassigned jobs that will allow us to find an upper bound on the increase in the cost.

Consider the machines whose configuration is $C$. For every $0 \le r \le y$, create $X'_C$ *bins* of size $t_r(C) \cdot \gamma \cdot (1+\delta)^{j_1(C)-1}$ each (these bins are called bins of the first kind), and $X'_C$ bins of size $\frac{3\gamma}{\delta} \cdot (1+\delta)^{j_1(C)-1}$ (these bins are called bins of the second kind); if $t_r(C) = 0$, then we introduce only the second kind of bins. The additional jobs are those that are packed into bins of the second kind. We define the allocation of jobs to machines by packing them as items into these bins. If the number of items of a certain type assigned using this packing process exceeds the existing number of jobs, then some of the spaces allocated in this process for items will not receive jobs. For every $i \in \mathcal{I}'(C)$, pack $Y'_{r,i,C}$ *items* of size $(1+\delta)^i$ into these bins using First Fit.

We show that all items are packed. Assume by contradiction that this is not the case. Since the size of each item is below $\gamma(1+\delta)^{j_1(C)-1}$ and there is an item that cannot be packed, each bin of the first kind (if such a bin exists) is occupied by at least $(t_r(C)-1)\cdot\gamma\cdot(1+\delta)^{j_1(C)-1}$, and each bin of the second kind is occupied by at least $(\frac{3}{\delta}-1)\cdot\gamma\cdot(1+\delta)^{j_1(C)-1}$. We find that the total size of items of density $(1+\delta)^r$ that are to be packed into these bins as small, which is equal to $\sum_{i\in\mathcal{I}'(C)}(1+\delta)^i Y'_{r,i,C}$ is above $(t_r(C)+\frac{3}{\delta}-2)\cdot\gamma\cdot(1+\delta)^{j_1(C)-1}\cdot X'_C$, contradicting the upper bound that we proved on $\sum_{i\in\mathcal{I}'(C)}(1+\delta)^i\cdot Y'_{r,i,C}$, since $\frac{3}{\delta}-2>2+\frac{2}{\delta}$ (since $\delta\le\frac{1}{8}$).

**Bounding the increase in the cost of a machine due to the first assignment step.** The increase in the cost of each machine (since the total size of small jobs of density $(1+\delta)^r$ for each $0\le r\le y$ may become larger) can be upper bounded as follows. A possible schedule, i.e., a permutation of the job set of the machine, is obtained by assigning the items of the bins of the first kind as a block for each value of $r$ of size at most $t_r(C)\cdot\gamma(1+\delta)^{j_1(C)-1}$, and the jobs of the bins of the second kind as a block of the last jobs assigned to the machine (this is possibly not an optimal ordering). For a job $j$ that is small for its configuration the difference between its cost and $\Gamma_j$ is already included in the objective function value (since it is assigned as a small job), and thus we compute the total $\Gamma$-values of the added blocks. Instead of considering these blocks (for different values of $r$) separately, we see it as one block assuming that all densities are equal to $(1+\delta)^y$ (this assumption cannot reduce the cost). We assume without loss of generality that $j_2(C)=0$. Let

$$W=\sum_{0\le r\le y, i\in\mathcal{I}(C)}(1+\delta)^i\cdot n_{r,i}(C)+\sum_{r=0}^{y}t_r(C)\cdot(\gamma\cdot(1+\delta)^{j_1(C)-1})$$

be the work that was used for calculating $cost(C)$ in $\mathcal{P}$. The total size of this united block of small jobs added to this machine is at most $\frac{3(y+1)\gamma}{\delta}(1+\delta)^{j_1(C)-1}$, and thus, the sum of $\Gamma$-values for all these blocks is at most

$$(1+\delta)^y\cdot\left(W+\frac{3\cdot(y+1)\cdot\gamma(1+\delta)^{j_1(C)-1}}{2\delta}\right)\cdot\frac{3(y+1)\gamma(1+\delta)^{j_1(C)-1}}{\delta}$$

by Lemma 3. We use $W\ge(1+\delta)^{j_1(C)-2}$, $\delta\le 1/8$, $y+1\le\frac{1}{\delta^9}$, $\gamma\le\delta^{1512}$, and $(1+\delta)^y\cdot\gamma=\delta^{12}$, and get

$$\frac{3(y+1)\gamma(1+\delta)^{j_1(C)-1}}{2\delta}\le\delta^{1501}\cdot W\ ,$$

and $(1+\delta)^y\cdot\frac{3(y+1)\gamma(1+\delta)^{j_1(C)-1}}{\delta}\le 3(1+\delta)\delta^2\cdot W$ and get that the total $\Gamma$-values of the added blocks is at most

$$(1+\delta^{1501})W\cdot 3(1+\delta)\cdot\delta^2\cdot W\le 7\delta^2\cdot W^2/2\ .$$

By Lemma 1, we have $cost(C)\ge W^2/2$ as all densities are at least 1. Thus, the second increase in the cost is by an additive factor of at most $7\delta^2\cdot Z^*$.

**Bounding the cost of scheduling the unassigned jobs.** Recall that $n'_{r,i}$ is the number of unassigned jobs of size $(1+\delta)^i$ and density $(1+\delta)^r$, and for $C \in C_H$, $X'_C = X^*_C$ and $Y'_{r,i,C} \geq Y^*_{r,i,C}$. Using (2), we have

$$\begin{aligned} n'_{r,i} &= n_{r,i} - \textstyle\sum_{C\in C_H} n_{r,i}(C)\cdot X'_C - \sum_{C\in C_H : i\in\mathcal{I}'(C)} Y'_{r,i,C} \\ &\quad - \textstyle\sum_{C\in C_L} n_{r,i}(C)\cdot X'_C - \sum_{C\in C_L : i\in\mathcal{I}'(C)} Y'_{r,i,C} \\ &\leq n_{r,i} - \textstyle\sum_{C\in C_H} n_{r,i}(C)\cdot X^*_C - \sum_{C\in C_H : i\in\mathcal{I}'(C)} Y^*_{r,i,C} \\ &\quad - \textstyle\sum_{C\in C_L}(n_{r,i}(C)\cdot(X^*_C-1)) - \sum_{C\in C_L : i\in\mathcal{I}'(C), X^*_C>0}(Y^*_{r,i,C} - \frac{Y^*_{r,i,C}}{X^*_C} - 1) \\ &= \textstyle\sum_{C\in C_L} n_{r,i}(C) + \sum_{C\in C_L : i\in\mathcal{I}'(C), X^*_C>0}(\frac{Y^*_{r,i,C}}{X^*_C}+1) \ . \end{aligned}$$

The total size of unassigned jobs is at most

$$\begin{aligned} &\textstyle\sum_{r=0}^{y}\sum_{C\in C_L}\sum_{i\in\mathcal{I}(C)}(1+\delta)^i\cdot n_{r,i}(C) \\ &+ \textstyle\sum_{r=0}^{y}\sum_{C\in C_L : X^*_C>0}\sum_{i\in\mathcal{I}'(C)}(1+\delta)^i\cdot(\frac{Y^*_{r,i,C}}{X^*_C}+1) \ . \end{aligned}$$

Using

$$\sum_{i\in\mathcal{I}'(C)}(1+\delta)^i\cdot\frac{Y^*_{r,i,C}}{X^*_C} \leq (t_r(C)+1)\cdot\gamma\cdot(1+\delta)^{j_1(C)-1}$$

for $C\in C_L$ such that $X^*_C>0$ (which holds by constraint (3) divided by $X^*_C$), the total size of unassigned jobs is at most

$$\begin{aligned} &\textstyle\sum_{r=0}^{y}\sum_{C\in C_L}\Big((t_r(C)+1)\gamma(1+\delta)^{j_1(C)-1} + \sum_{i\in\mathcal{I}(C)}(1+\delta)^i n_{r,i}(C)\Big) \\ &\qquad + \textstyle\sum_{r=0}^{y}\sum_{C\in C_L}\sum_{i\in\mathcal{I}'(C)}(1+\delta)^i \ . \end{aligned}$$

Using

$$\sum_{0\leq r\leq y, i\in\mathcal{I}(C)}(1+\delta)^i n_{r,i}(C) + \sum_{r=0}^{y} t_r(C)(\gamma(1+\delta)^{j_1(C)}) \leq (1+\delta)^{j_1(C)+1}$$

(by the definition of a configuration) and $\sum_{i\in\mathcal{I}'(C)}(1+\delta)^i \leq \gamma(1+\delta)^{j_1(C)}/\delta$ for any $C\in\mathcal{C}$, the total size of unassigned jobs is at most

$$\begin{aligned} &\textstyle\sum_{C\in C_L}(1+\delta)^{j_1(C)+1} + \sum_{r=0}^{y}\sum_{C\in C_L}\gamma(1+\delta)^{j_1(C)-1} \\ &+ \textstyle\sum_{r=0}^{y}\sum_{C\in C_L}\gamma(1+\delta)^{j_1(C)}/\delta < \sum_{C\in C_L} 2(1+\delta)^{j_1(C)} \ . \end{aligned}$$

since $(1+\delta+(y+1)\gamma(1/\delta+1)) \leq 1+\delta+\frac{1}{\delta^9}\cdot\delta^{1500}\cdot\frac{2}{\delta} < 2$.

The proof of the following lemma is derived using the definition of $g(\delta)$.

▶ **Lemma 11.** *The total size of unassigned jobs is at most* $2\gamma$.

Now we show that this leads to the required result. The jobs are assigned to a machine with a configuration $C'$ such that $j_1(C')\in\{1,0,-1\}$. Let $x$ be the completion time of this machine before modifications, where $x\in(\frac{1}{(1+\delta)^3},(1+\delta)^2]$. The modifications described so far during the first assignment step may increase the completion time of the machine by at most

$$\frac{3}{\delta}\cdot\gamma(y+1) < \delta^{1400}$$

(since $3\gamma(y+1)/\delta \leq \frac{3}{\delta} \cdot \delta^{1500}\frac{1}{\delta^9} < \delta^{1400}$). Even if the density of each such job is $(1+\delta)^y$, and the unassigned jobs are now assigned as a block on the most loaded machine of speed 1 starting at time $x + \delta^{1400}$, the sum of their weighted completion times will be at most

$$2\gamma(1+\delta)^y \cdot (x + \delta^{1400} + 2\gamma) < 2\delta^{12}(x + \delta^{1399}) \ ,$$

while the value of $cost(C')$ is at least $x^2/2$. It holds that

$$\frac{2\delta^{12}(x+\delta^{1399})}{x^2/2} = \frac{4\delta^{12}}{x} + \frac{4\delta^{1411}}{x^2} \leq 4\delta^{12}(1+\delta)^3 + 4\delta^{1411}(1+\delta)^6 < 6\delta^{12} \ ,$$

so the increase in the cost is at most by an additive factor of $15\delta^{12} \cdot Z^*$. The total cost will be at most $Z^*(1 + \delta^{100} + 7\delta^2 + 6\delta^{12}) < (1+\delta) \cdot Z^*$, and thus the following theorem holds.

▶ **Theorem 12.** *There exists a schedule whose cost is at most $(1+\delta) \cdot Z^*$, and such a schedule can be constructed in polynomial time from $(X^*, Y^*)$.*

**Summary of the algorithm for the bounded ratio problem.** In summary, in this section we provided an $(1+\delta)$-approximation algorithm for the bounded ratio scheduling problem. The algorithm tests all possible intervals $D_{j,b}$, and for each one it constructs the MILP $\mathcal{P}$, and finds an optimal solution. The best solution is converted into a schedule. We conclude the paper with the statement of our main result.

▶ **Theorem 13.** *There is an EPTAS for the problem of minimizing the total weighted completion time on uniformly related machines.*

## References

1 Foto N. Afrati, Evripidis Bampis, Chandra Chekuri, David R. Karger, Claire Kenyon, Sanjeev Khanna, Ioannis Milis, Maurice Queyranne, Martin Skutella, Clifford Stein, and Maxim Sviridenko. Approximation schemes for minimizing average weighted completion time with release dates. In *Proc. 40th Symp. Foundations of Computer Science (FOCS)*, pages 32–44, 1999. doi:10.1109/SFFCS.1999.814574.

2 Foto N. Afrati and Ioannis Milis. Designing PTASs for MIN-SUM scheduling problems. *Discrete Applied Mathematics*, 154(4):622–639, 2006. doi:10.1016/J.DAM.2005.05.014.

3 N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. Approximation schemes for scheduling. In *Proc. 8th Symp. on Discrete Algorithms (SODA)*, pages 493–500, 1997.

4 N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1(1):55–66, 1998.

5 Alok Baveja, Xiaoran Qu, and Aravind Srinivasan. Approximating weighted completion time via stronger negative correlation. *Journal of Scheduling*, 27(4):319–328, 2024. doi:10.1007/S10951-023-00780-Y.

6 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 *Proc. 34th International Symp. on Algorithms and Computation (ISAAC)*, 2023. doi:10.4230/LIPIcs.ISAAC.2023.13.

7 M. Cesati and L. Trevisan. On the efficiency of polynomial time approximation schemes. *Information Processing Letters*, 64(4):165–171, 1997. doi:10.1016/S0020-0190(97)00164-6.

8 Chandra Chekuri and Sanjeev Khanna. A PTAS for minimizing weighted completion time on uniformly related machines. In *Proc. 28th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 848–861, 2001. doi:10.1007/3-540-48224-5_69.

9 R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer-Verlag, Berlin, 1999.

10 L. Epstein and J. Sgall. Approximation schemes for scheduling on uniformly related and identical parallel machines. *Algorithmica*, 39(1):43–57, 2004. doi:10.1007/S00453-003-1077-7.

11 Leah Epstein and Asaf Levin. An efficient polynomial time approximation scheme for load balancing on uniformly related machines. *Mathematical Programming*, 147(1-2):1–23, 2014. doi:10.1007/S10107-013-0706-4.

12 Leah Epstein and Asaf Levin. Minimum total weighted completion time: Faster approximation schemes. *CoRR*, abs/1404.1059, 2014. arXiv:1404.1059.

13 Leah Epstein, Asaf Levin, Alan J. Soper, and Vitaly A. Strusevich. Power of preemption for minimizing total completion time on uniform parallel machines. *SIAM Journal on Discrete Mathematics*, 31(1):101–123, 2017. doi:10.1137/16M1066610.

14 J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer-Verlag, Berlin, 2006.

15 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

16 Stanisław Gawiejnowicz. A review of four decades of time-dependent scheduling: Main results, new topics, and open problems. *Journal of Scheduling*, 23(1):3–47, 2020. doi:10.1007/S10951-019-00630-W.

17 M. X. Goemans. Improved approximation algorithms for scheduling with release dates. In *Proc. 8th Symp. on Discrete Algorithms (SODA)*, pages 591–598, 1997.

18 D. S. Hochbaum. Various notions of approximations: Good, better, best, and more. In D. S. Hochbaum, editor, *Approximation algorithms.* PWS Publishing Company, 1997.

19 D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*, 32(1):130–136, 1985. doi:10.1145/2455.214106.

20 D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM*, 34(1):144–162, 1987. doi:10.1145/7531.7535.

21 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(3):539–551, 1988. doi:10.1137/0217033.

22 W. A. Horn. Minimizing average flow time with parallel machines. *Operations Research*, 21(3):846–847, 1973.

23 K. Jansen. An EPTAS for scheduling jobs on uniform processors: Using an MILP relaxation with a constant number of integral variables. *SIAM Journal on Discrete Mathematics*, 24(2):457–485, 2010. doi:10.1137/090749451.

24 K. Jansen and C. Robenek. Scheduling jobs on identical and uniform processors revisited. In *Proc. of the 9th International Workshop on Approximation and Online Algorithms (WAOA)*, pages 109–122, 2011.

25 Klaus Jansen, Kim-Manuel Klein, and José Verschae. Closing the gap for makespan scheduling via sparsification techniques. *Mathematics of Operations Research*, 45(4):1371–1392, 2020. doi:10.1287/MOOR.2019.1036.

26 Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *Proc. 15th Symp. Theory of Computing (STOC)*, pages 193–206, 1983. doi:10.1145/800061.808749.

27 Dušan Knop and Martin Koutecký. Scheduling meets n-fold integer programming. *Journal of Scheduling*, 21:493–503, 2018. doi:10.1007/S10951-017-0550-0.

28 Ishai Kones and Asaf Levin. A unified framework for designing eptas for load balancing on parallel machines. *Algorithmica*, 81(7):3025–3046, 2019. doi:10.1007/S00453-019-00566-9.

29 Hanane Krim, Nicolas Zufferey, Jean-Yves Potvin, Rachid Benmansour, and David Duvivier. Tabu search for a parallel-machine scheduling problem with periodic maintenance, job rejection and weighted sum of completion times. *Journal of Scheduling*, pages 1–17, 2022.

30 H. W. Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983. `doi:10.1287/MOOR.8.4.538`.

31 D. Marx. Parameterized complexity and approximation algorithms. *The Computer Journal*, 51(1):60–78, 2008. `doi:10.1093/COMJNL/BXM048`.

32 Sartaj K. Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM*, 23:116–127, 1976. `doi:10.1145/321921.321934`.

33 Martin Skutella and Gerhard J. Woeginger. A PTAS for minimizing the total weighted completion time on identical parallel machines. *Mathematics of Operations Research*, 25(1):63–75, 2000. `doi:10.1287/MOOR.25.1.63.15212`.

34 W. E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3:59–66, 1956.

35 Vipin Ravindran Vijayalakshmi, Marc Schröder, and Tami Tamir. Minimizing total completion time with machine-dependent priority lists. *European Journal of Operational Research*, 315(3):844–854, 2024. `doi:10.1016/J.EJOR.2023.12.030`.