An efficient polynomial time approximation scheme for load balancing on uniformly related machines

Leah Epstein · Asaf Levin

Abstract We consider basic problems of non-preemptive scheduling on uniformly related machines. For a given schedule, defined by a partition of the jobs into $m$ subsets corresponding to the $m$ machines, $C_i$ denotes the completion time of machine $i$. Our goal is to find a schedule that minimizes or maximizes $\sum_{i=1}^{m} C_i^p$ for a fixed value of $p$ such that $0 < p < \infty$. For $p > 1$ the minimization problem is equivalent to the well-known problem of minimizing the $\ell_p$ norm of the vector of the completion times of the machines, and for $0 < p < 1$, the maximization problem is of interest. Our main result is an efficient polynomial time approximation scheme (EPTAS) for each one of these problems. Our schemes use a non-standard application of the so-called shifting technique. We focus on the work (total size of jobs) assigned to each machine and introduce intervals of work that are forbidden. These intervals are defined so that the resulting effect on the goal function is sufficiently small. This allows the partition of the problem into sub-problems (with subsets of machines and jobs) whose solutions are combined into the final solution using dynamic programming. Our results are the first EPTAS’s for this natural class of load balancing problems.

Keywords EPTAS · Load balancing · Scheduling · Approximation algorithms

Mathematics Subject Classification 68W25 Approximation algorithms · 68W40 Analysis of algorithms · 68Q25 Analysis of algorithms and problem complexity · 90C27 Combinatorial optimization · 90C59 Approximation methods and heuristics
1 Introduction

We consider non-preemptive scheduling problems on \( m \) uniformly related machines. In such problems, we are given a set of jobs \( \{1, 2, \ldots, n\} \), where each job \( j \) has a positive size \( p_j \). The jobs need to be partitioned into \( m \) subsets \( S_1, \ldots, S_m \), with \( S_i \) being the subset of jobs assigned to machine \( i \). We let \( s_i \) denote the speed of machine \( i \), and the processing of a job \( j \) requires \( \frac{p_j}{s_i} \) time units, if \( j \) is assigned to machine \( i \). For such a solution (also known as a schedule), we let \( C_i = \sum_{j \in S_i} \frac{p_j}{s_i} \) be the completion time (or load) of machine \( i \). The work of machine \( i \) (also called the weight of the machine) is \( W_i = \sum_{j \in S_i} p_j = C_i \cdot s_i \), that is, the total size of the jobs that are assigned to \( i \). The special case of identical machines is where \( s_i = 1 \) for all \( i \). The makespan of the schedule is \( \max_i C_i \), and the optimization problem of finding a schedule that minimizes the makespan is well-studied (see e.g. \([19, 20, 23, 24, 26, 27]\)). The problem of finding a schedule that maximizes \( \min_i C_i \) is the well-known Santa Claus problem on uniformly related machines (see e.g. \([2, 5, 8, 15, 18, 32]\)). These problems are both concerned with the optimization of the extremum values of the set \( \{C_1, \ldots, C_m\} \).

Motivated by minimizing average latency in storage allocation applications (rather than worst-case latency), researchers have suggested to study the optimization goal of minimizing the \( \ell_2 \) norm (and the goal of minimizing the \( \ell_p \) norm for \( p > 1 \)) of the vector of completion times of the machines (see e.g. \([3, 4, 11, 12, 30]\)). It was stated more recently by Bansal and Pruhs \([7]\) that: “The standard way to compromise between optimizing for the average and optimizing for the worst case is to optimize the \( \ell_p \) norm, generally for something like \( p = 2 \) or \( p = 3 \).” An additional perspective of using the \( \ell_p \) norm as an objective function has arisen recently in algorithmic game theory \([9]\). Note that the minimization of the \( \ell_p \) norm is equivalent to minimizing the sum of the \( p \)-th powers of the completion times of machines. Thus, we consider objective functions in which the entire vector \( C = (C_1, \ldots, C_m) \) affects the value of the objective function. Our class of objective functions includes the minimization of the sum of the \( p \)-th powers of the completion times of machines, which is equivalent to the minimization of the \( \ell_p \) norm of \( C \). More precisely, given a fixed real number \( p \) such that \( 0 < p < \infty \), we consider the problem of minimizing \( \sum_{i=1}^{m} C_i^p \) and the problem of maximizing \( \sum_{i=1}^{m} C_i^p \). The minimization problem for \( p \leq 1 \) is trivially solved by placing all the jobs on one of the fastest machines. Therefore, we consider the minimization problem only for values of \( p \) such that \( p > 1 \). Similarly, the maximization problem is trivially solved for \( p \geq 1 \) by placing all the jobs on one of the slowest machines. Hence, we consider the maximization problem only for values of \( p \) such that \( p < 1 \).

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. An \( \mathcal{R} \)-approximation algorithm for a maximization problem is a polynomial time algorithm that always finds a feasible solution of value at least \( \frac{1}{\mathcal{R}} \) times the value of an optimal solution (note that we use the convention of approximation ratios greater than 1 for maximization problems). 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 \(h(\varepsilon) \cdot \text{poly}(n)\) where \(h\) is some (not necessarily polynomial) function and \(\text{poly}(n)\) is a polynomial function of the length of the (binary) encoding of the input. We do not include \(m\) in the running time as in scheduling problems such as those studied here it can be assumed without loss of generality that \(m \leq n\) (see Sect. 2). A fully polynomial time approximation scheme (FPTAS), is a stronger concept, defined like an EPTAS, but the function \(h\) 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 \(h(\varepsilon) \cdot \text{poly}(n)\). Note that whereas a PTAS may have time complexity of the form \(n^{g(\varepsilon)}\), where \(g\) is for example linear or even exponential function of \(\frac{1}{\varepsilon}\), 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 \([10,13,17,31]\)).

Our main result is a class of EPTAS’s for minimizing \(\sum_{i=1}^{m} C_i^p\) for any fixed value of \(p > 1\), and for the problem of maximizing \(\sum_{i=1}^{m} C_i^p\) for any fixed positive value of \(p < 1\). Note that these problems are known to be strongly NP-hard even for identical machines (via the standard reduction from the 3- PARTITION problem) and therefore our results are the best possible (since FPTAS’s do not exist for these problems unless \(P=NP\)). Our results are the first EPTAS’s for these important load balancing problems on uniformly related machines. Note that the running time of an approximation scheme for a scheduling or load balancing problem is expected to be polynomial in the number of jobs as well as in the number of machines, while for fixed (constant) numbers of machines, load balancing problems typically have FPTAS’s \([6,14,16,25]\).

We next review the previous PTAS and EPTAS results for an arbitrary (non-constant) number of uniformly related machines and the special case of identical machines (where all machines have unit speeds). It was shown by Hochbaum and Shmoys \([23,24]\) that the makespan minimization problem has a PTAS for identical machines and for uniformly related machines. It was noted in \([21]\) that the PTAS of \([23]\) for identical machines can be converted into an EPTAS by using integer program in fixed dimension instead of dynamic programming. Recently, Jansen \([26]\) was able to solve the long-standing open problem of establishing an EPTAS for the makespan minimization problem on uniformly related machines (see \([27]\) for an alternative EPTAS for this problem). The Santa Claus problem is also known to have a PTAS and an EPTAS for identical machines \([2,32]\). For uniformly related machines a PTAS is known \([5,15]\).

The problems studied here are known to have EPTAS’s for identical machines \([1,2]\), and PTAS’s for uniformly related machines \([15]\). The existence of EPTAS’s for these problems on uniformly related machines was stated as an open problem by \([15]\).\(^1\) This open problem is resolved in our work.

\(^1\) The statement of the open problem did not contain the word EPTAS, but the discussion there explains that the nice property of the PTAS’s of \([1,2]\) is that the function of \(\varepsilon\) does not appear in the exponent of \(n\) in the running time.
1.1 Outline

Our EPTAS’s have the following structure. First, we sort the machines by a non-decreasing order of their weights in an optimal solution (according to either non-increasing or non-decreasing speed). We note that some machines may get zero weights; we guess the number of such machines and remove these machines from the instance. We round the processing times of the jobs and the speeds of the machines, so that the number of possible values is reduced sufficiently, and so that all job sizes are integer multiples of a small value.

Next, we observe that we can extend the EPTAS for identical machines to the case where we are guaranteed that in an optimal solution, the ratio between the maximum work of any machine and the minimum work of any machine is bounded from above. We show that this case can occur only if the speed ratio is bounded from above as well. We extend this EPTAS further to allow some total size of jobs (within a given budget) to remain unscheduled. This will be our building block in the design of the EPTAS for the general case.

To reduce the general case into a series of sub-problems of the former type, where in each sub-problem a certain subsequence of the machines is considered, we create gaps in the set of allowed weights of machines in the sense that there will be intervals of weights that no machine is allowed to have. For that, we apply the so-called shifting technique [22] in an original way. Afterwards, we apply dynamic programming to determine the series of sub-problems, that is, the subsets of machines whose weights come from each interval of allowed weights. The EPTAS for the special case is used as a black box in this dynamic programming, where unscheduled jobs of one sub-problem are scheduled later by another sub-problem.

2 Preliminaries

In this paper we consider the sum of the \( p \)-th powers of a vector rather than the \( \left( \frac{1}{p} \right) \)-th power of this value. Note that since \( p \) is a fixed constant, our results apply also for this last alternative measure (which is the \( \ell_p \) norm for the case \( p > 1 \)). Throughout the paper, for a solution \( A \) we denote by \( A \) both the solution and the value of the objective function for this solution.

When we consider the maximization problem, we sometimes allow the algorithm to avoid assigning some of the jobs. It is clear that adding these jobs arbitrarily to the schedule can only improve the solution. Hence, if we can bound the value of the objective function for the solution that assigns a subset of the jobs, after adding the unscheduled jobs (to create a complete solution), we get (at least) the same performance guarantee.

Let \( \varepsilon \) be a small constant such that \( 0 < \varepsilon < \frac{1}{2} \) and \( \frac{1}{\varepsilon} > 2 \) is an integer. Epstein and Sgall [15] observed the following claim.

**Claim 1** Let \( i_1 \) and \( i_2 \) be a pair of machines such that \( s_{i_1} < s_{i_2} \), that is, \( i_2 \) is strictly faster than \( i_1 \). For the minimization problem with \( p > 1 \), any optimal solution satisfies \( W_{i_1} \leq W_{i_2} \). For the maximization problem with \( p < 1 \), any optimal solution satisfies \( W_{i_1} \geq W_{i_2} \).
Motivated by the above claim we will sort the machines according to their weights. That is, when we consider the minimization problem we will assume that \( s_1 \leq s_2 \leq \cdots \leq s_m \), whereas when we consider the maximization problem we will assume that \( s_1 \geq s_2 \geq \cdots \geq s_m \). In this way, machines of lower indices should get smaller weight than machines with higher indices (or equal weight). For machines with equal speeds, we assume that a machine appearing later in the ordering cannot receive smaller work than a machine appearing earlier in the ordering, and this can be achieved by renaming the machines in an optimal solution. Our assumption \( m \leq n \) is justified by this claim as well, since if \( m > n \), then it can be assumed that machines \( 1, 2, \ldots, m - n \) receive weight zero in optimal solutions, and can be removed. We next consider a pair of machines \( i < i' \) such that the speed ratio between the speed of the faster machine of the two machines and the speed of the slower machine of the two is significantly larger than 1. We already know that \( W_i \leq W_{i'} \), and our next goal is to strengthen this bound. Let \( \delta \) be such that \( 0 < \delta \leq \varepsilon \).

**Lemma 2** Consider the minimization problem \((p > 1)\). The function \( \alpha(\delta) = \frac{\delta}{p} \) satisfies the following property. For any pair of machines \( i < i' \) such that \( W_i > 0 \), if \( s_i \leq \alpha(\delta) \cdot s_{i'} \), then in any optimal solution \( W_i < \delta W_{i'} \).

**Proof** Assume by contradiction that \( W_i \geq \delta W_{i'} \) in some optimal solution (and thus \( W_i > 0 \)). The part of the cost of the solution that is incurred by the machines \( i, i' \) is at least \( \left( \frac{W_i}{s_i} \right)^p \). We modify the solution by moving all the jobs that were scheduled on machine \( i \) to machine \( i' \). It suffices to show that the completion time of machine \( i' \) in this new solution is smaller than \( \frac{W_i}{s_i} \) (since if the part of the cost of the modified solution that is incurred by the machines \( i, i' \) is smaller than its value before the change, then this contradicts optimality). The last claim holds because the completion time of \( i' \) in the new solution is \( \frac{W_i + W_{i'}}{s_{i'}} < \frac{2W_i}{\delta s_{i'}} \leq \frac{2W_i}{\delta s_i} \cdot \alpha(\delta) = \frac{W_i}{s_i} \) where the first inequality holds because \( W_{i'} \leq \frac{W_i}{\delta} \) and \( \delta < 1 \) and \( W_i > 0 \), the second inequality holds because \( s_i \leq \alpha(\delta) \cdot s_{i'} \), and the equality holds by the definition of \( \alpha(\delta) \).

**Lemma 3** Consider the maximization problem \((p < 1)\). The function \( \alpha(\delta) = ((1 + \delta)^p - 1)^{1/p} \) satisfies the following property. For any pair of machines \( i < i' \) such that \( W_{i'} > 0 \), if \( s_{i'} \leq \alpha(\delta) \cdot s_i \), then in any optimal solution \( W_i < \delta W_{i'} \).

**Proof** Assume by contradiction that the claim does not hold, that is, there exist such machines \( i \) and \( i' \) with \( W_i \geq \delta W_{i'} \). Let \( \delta' = \frac{W_i}{W_{i'}} \), where \( \delta \leq \delta' \leq 1 \) as \( W_i \leq W_{i'} \). We compare the current solution with a new solution that schedules all the jobs (which were previously scheduled on either \( i \) or \( i' \)) on machine \( i' \). It suffices to show that this new solution is better, that is, that the following inequality holds: \( \left( \frac{W_i + W_{i'}}{s_{i'}} \right)^p > \left( \frac{W_i}{s_i} \right)^p + \left( \frac{W_{i'}}{s_{i'}} \right)^p \). This inequality is equivalent to \( \left( \frac{1 + \delta' s_i}{s_{i'}} \right)^p > \left( \frac{\delta'}{s_{i'}} \right)^p + \left( \frac{1}{s_{i'}} \right)^p \) that we will prove.

We have \( \left( \frac{\delta'}{s_{i'}} \right)^p \leq \left( \frac{\delta' - \alpha(\delta)}{s_{i'}} \right)^p = \left( \frac{s_i}{s_{i'}} \right)^p \cdot ((1 + \delta)^p - 1) \), using \( s_{i'} \leq \alpha(\delta) \cdot s_i \) and the definition of \( \alpha(\delta) \). Therefore, it is sufficient to prove \((1 + \delta')^p > (\delta')^p ((1 + \delta)^p - 1) + 1 \). If \( \delta' = 1 \), the inequality holds since \( 2^p > (1 + \delta)^p \), since \( \delta < 1 \) and as \( x^p \) is a strictly monotone increasing function of \( x \) (for \( 0 \leq x < \infty \) and any \( p > 0 \)). Otherwise using
0 < δ ≤ δ' < 1 we have \((δ′)^p(1 + δ)^p - 1) + 1 < ((1 + δ)^p - 1) + 1 ≤ (1 + δ′)^p\), proving the last claim.

Note that in both cases, \(\alpha(δ) ≤ δ ≤ ε\). This is clear for the minimization problem, and for the maximization problem it holds because \((\alpha(δ))^p + 1 = (1 + δ)^p ≤ 1^p + δ^p = δ^p + 1\) where the inequality holds by the concavity of \(x^p\) for \(p < 1\), and the claim holds by the monotonicity of \(x^{1/p}\). Moreover, the function \(\alpha(δ)\) is a monotonically increasing function of \(δ\) in both cases.

We summarize the last two lemmas by the following straightforward corollary, which we will use for the analysis of subsets of machines with relatively similar work. The corollary allows us to assume that in this case the speeds of all machines in the subset are sufficiently close. Let \(λ(ε)\) be a monotonically non-increasing function of \(ε\) (i.e., it becomes large as \(ε\) becomes small) such that \(λ(ε)\) is a positive integer for any valid value of \(ε\) (the exact function is fixed later), and let \(γ(ε) = \frac{1}{ε^λ(ε)}\). Thus \(γ(ε)\) is an integer for any valid value of \(ε\). Let \(β(ε) = 1/α(\frac{1}{γ(ε)}))\). Note that in both cases, \(β(ε)\) is a monotonically decreasing function of \(ε\), and we have \(β(ε) = \frac{2}{ε^α(ε)}\) for the minimization problem, and \(β(ε) = \left(\frac{1}{(1 + ε^{λ(ε)})^{p-1}}\right)^{1/p}\) for the maximization problem.

**Corollary 4** Consider an optimal solution, and a pair of machines \(i\) and \(i'\) such that \(i < i'\). If \(W_i ≥ \frac{Wi}{Y(ε)} > 0\), then the ratio between the speeds \(\max\{s_i, s_{i'}\}\) and \(\min\{s_i, s_{i'}\}\) is at most \(β(ε)\).

**Proof** Since \(\frac{1}{γ(ε)} ≤ ε\), we can apply Lemmas 2, 3 with \(δ = \frac{1}{γ(ε)}\). We find that if \(W_i ≥ \frac{Wi}{γ(ε)}\), then \(\frac{s_{i'}}{s_i} ≤ \frac{1}{α(δ)} = β(ε)\) for the minimization problem, and \(\frac{s_i}{s_{i'}} ≤ \frac{1}{α(δ)} = β(ε)\) for the maximization problem. □

For any input \(\hat{I}\), we let \(\text{SOL}(\hat{I})\) denote the value (cost or profit) of the solution \(\text{SOL}\). An optimal algorithm is denoted by \(\text{OPT}\). We let \(J\) denote an input (a set of machines and jobs) for one of the problems, and modify it as follows.

**2.1 First rounding step**

In what follows we would like to use the property that the speeds are integer powers of \(1 + ε\), and we create a modified instance that satisfies this property. Denote by \(J\) the original instance. The instance \(J'\) is created from \(J\) by increasing the speed of each machine of \(J\) to the next value of the form \(1 + ε)^j\) (for an integer \(j\)). Moreover, the instance \(J''\) is created from \(J'\) by increasing the size of each job to the next value of the form \(1 + ε)^j\) (for an integer \(j\)). Since these three instances have the same machines and jobs (with slightly different speeds and sizes, respectively), any solution for one of them immediately gives solutions for the others. We will see these solutions as the same solution (since the assignment function of jobs to machines is the same).

This modification is justified by the following observation.

**Claim 5** Any solution \(\text{SOL}(J')\) satisfies \(\text{SOL}(J') ≤ \text{SOL}(J) ≤ (1 + ε)^p \cdot \text{SOL}(J')\) and \(\text{SOL}(J') ≤ \text{SOL}(J'') ≤ (1 + ε)^p \cdot \text{SOL}(J').\) Thus \(\frac{\text{SOL}(J'')}{(1 + ε)^p} ≤ \text{SOL}(J) ≤ (1 + ε)^p \cdot \text{SOL}(J'').\) Additionally, \(\frac{\text{OPT}(J)}{(1 + ε)^p} ≤ \text{OPT}(J'') ≤ (1 + ε)^p \cdot \text{OPT}(J).\)
Proof  The first claim holds since by increasing the speed of a machine, the completion
time of this machine may decrease by a multiplicative factor of at most $1 + \varepsilon$ and it
cannot increase. The second claim holds since by increasing the sizes of jobs by a
multiplicative factor of at most $1 + \varepsilon$, the completion time of each machine may
increase by a multiplicative factor of at most $1 + \varepsilon$ and it cannot decrease.

Next, we prove the claim regarding optimal solutions. For the minimization prob-
lem, an optimal solution for $J'$ gives a solution for $J$ with a cost that is larger by at
most a multiplicative factor of $(1 + \varepsilon)p$, and an optimal solution for $J$ gives a solution
for $J'$ whose cost is at most $\text{OPT}(J)$. Moreover, an optimal solution for $J''$ gives a solution
for $J'$ of cost at most $\text{OPT}(J'')$, and an optimal solution for $J'$ gives a solution
for $J''$ whose cost is larger by at most a multiplicative factor of $(1 + \varepsilon)p$. For the
maximization problem, an optimal solution for $J'$ gives a solution for $J$ with at least
the same profit, and an optimal solution for $J$ gives a solution for $J'$ whose profit is
smaller by a multiplicative factor of at most $(1 + \varepsilon)p$. Moreover, an optimal solution
for $J''$ gives a solution for $J'$ of a profit that is no smaller than $\frac{\text{opt}(J'')}{(1+\varepsilon)p}$, and an optimal
solution for $J'$ gives a solution for $J''$ whose profit is not smaller. \hfill \Box

The value of $\varepsilon$ will be scaled later, and we will use the next assumption in the
building blocks of the algorithm.

Assumption 6  The speed of each machine as well as the size of each job are integer
powers of $1 + \varepsilon$.

2.2 Second rounding step

Let $p_j'$ be the size of job $j$ in the instance $J''$. Let $p_{\text{max}} = \max_{j=1,2,...,n} p_j'$. We now
define the final rounding step. The resulting instance will be called $I$, and we will
let $\text{OPT}$ denote a fixed optimal solution for the resulting instance after this step. For
the minimization problem, we apply the following rounding (down) of the (already
rounded) processing times for $J''$. If $p_j' \leq \frac{\varepsilon p_{\text{max}}}{n}$, then we round $p_j'$ down to be zero
and remove it from the instance. Otherwise, we round $p_j'$ down to the next integer
multiple of $\mu = \frac{\varepsilon^2 p_{\text{max}}}{n}$. With a slight abuse of notation, if $j$ remains a part of the
instance, then we let this rounded size be denoted by $p_j$. Since $\frac{\varepsilon p_{\text{max}}}{n} = \mu \frac{\varepsilon}{\varepsilon}$, and $\frac{1}{\varepsilon}$ is an integer, we find that if $j$ is a part of $I$, then $p_j \geq \frac{\mu}{\varepsilon}$, and otherwise $p_j = 0$. We claim
that the instance $I$ also has the property that its largest job has size $p_{\text{max}}$. Obviously,
since $\varepsilon \leq \frac{1}{2}$ and $n \geq 1$, no such jobs were removed from the instance. Moreover, $p_{\text{max}}$
is an integer multiple of $\mu$, since $\frac{p_{\text{max}}}{\mu} = \frac{n}{\varepsilon^2}$ is an integer (as $\frac{1}{\varepsilon}$ is an integer).

We say that a solution SOL for $I$ and a solution SOL' for $J''$ are related if every job
that exists in both instances (possibly with slightly different sizes) is assigned to the
same machine in both solutions, and there is a machine $i$ in SOL' that has a job of size
$p_{\text{max}}$, such that all jobs of $J'' \setminus J$ are assigned to it. Given a solution SOL for $I$, we can
create the related solution SOL' for $J''$, by assigning jobs exactly as in SOL (with the
sizes as in $J''$), if they also exist in $I$, and by assigning all the removed jobs (jobs that
exist in $J''$ but not in $I$) to a machine that has a job of size $p_{\text{max}}$ assigned to it (breaking
ties arbitrarily). Similarly, a solution SOL' for $J''$ can be converted into a solution for

\begin{flushright}
\textcopyright Springer
\end{flushright}
I by changing the sizes of jobs, and removing jobs of zero size. Note that we applied two separate rounding steps on the sizes of jobs. The goal of the first rounding step is to reduce the number of possible sizes, and the goal of the second rounding step is to obtain convenient job sizes.

**Lemma 7** Consider the minimization problem. Given a solution sol for I and a solution sol′ for J′′, such that the two solutions are related, we have sol(I) ≤ sol′(J′′) ≤ (1 + ε)opt(I). Moreover, opt(I) ≤ opt(J′′).

*Proof* Since we only round down sizes of jobs, sol(I) ≤ sol′(J′′) and opt(I) ≤ opt(J′′). To prove the second part, let i be the machine that has all jobs of J′′ \ I in sol′. For a machine i′ ≠ i, the only difference between the solutions are job sizes. We compute the effect of increasing the size of every job j assigned to i′ from pj to p′ j. For each job j that belongs to I, p′ j ≥ μ + ε and p′ j > p j − μ must hold, so we have  

\[
\frac{p_{j} - p_{j}'}{p_{j}} < \frac{\mu}{p_{j}} \leq \frac{\mu}{\mu / \varepsilon} = \varepsilon.
\]

Thus, for a machine i′ ≠ i, whose work in sol is υ, its work in sol′ can be larger by an additive factor of at most ευ compared to sol. As for machine i, let its work in sol be υ′. Since a job of size p_{max} is assigned to i′ in sol′, \(\varepsilon \cdot p_{max} \) compared to its size in sol (if it is not a part of I, then for this statement we see its size in I as zero). Thus, as there are at most n jobs assigned to i in sol′, the work of i is larger by at most an additive factor of ϵp_{max} compared to sol. We have \(\frac{\varepsilon \cdot p_{max}}{p_{max}} = \varepsilon\). Thus, the work of i in sol′ is at most (1 + ε)υ′. 

For the maximization problem, the second rounding step is defined as follows. In this case we let \(\mu = \frac{\varepsilon \cdot p_{max}}{n \cdot m \cdot \frac{p}{2}}\), and we round the processing time of every job down to an integer multiple of \(\mu\). Once again, J′′ is the instance before the rounding, where the size of job j is p′ j, and I is the instance after the rounding, where the size of job j is \(p_{j}\). In this case the two instances have the same jobs, so we can consider one solution sol and compare its cost for the two inputs. In this case the instance I does not necessarily have the property that its largest job has size \(p_{max}\), but the input I contains a job of size above \(p_{max} - \mu\) and the size of each job is at most \(p_{max}\). In this case, I may contain jobs of size zero, but such jobs will be neglected and will not be assigned by that algorithm. Thus, in the design of the algorithm, we assume that all jobs of I have sizes of at least \(\mu\).

**Observation 8** The definition of \(\mu\) satisfies \(\frac{\mu}{p_{max}} \geq \frac{\varepsilon}{n \cdot m \cdot \frac{p}{2}}\) in both cases.

**Lemma 9** Consider the maximization problem and a solution sol for J′′ and I. We have \((1 - \varepsilon)\text{sol}(J''') \leq \text{sol}(I) \leq \text{sol}(J'')\). Moreover, \(\text{opt}(I) \geq (1 - \varepsilon)\text{opt}(J''')\).

*Proof* Due to the rounding, \(\text{sol}(I) \leq \text{sol}(J''')\). Consider a machine i whose work in sol for J′′ denoted by \(\nu\). If \(\nu \leq n \mu\), then its work in the solution for I is obviously at least 0. Otherwise, since at most \(n\) jobs are assigned to i, its work in the solution for I is at least \(\nu - n \mu\). Consider the case where \(\nu > n \mu\). Then, the contribution of machine
An efficient polynomial time approximation scheme

An efficient polynomial time approximation scheme

Consider the case where \( v \leq n\mu \). Then, the contribution of machine \( i \) to \( \text{SOL}(I) \) is at least
\[
0 \geq \left( \frac{v}{s_i} \right)^p - \left( \frac{n\mu}{s_i} \right)^p = \left( \frac{v}{s_i} \right)^p - \frac{p_{\text{max}}}{m \cdot s_m} \geq \left( \frac{v}{s_i} \right)^p - \frac{p_{\text{max}}}{m \cdot s_m},
\]
where the second inequality holds because \( \text{SOL}(J'') \geq \left( \frac{p_{\text{max}}}{s_m} \right)^p \), which is the value of a solution that places the largest job on the slowest machine, and does not assign any of the other jobs.

Note that after the second rounding step, for both problems, the size of any job is an integer multiple of \( \mu \).

**Lemma 10** Given an interval \([L, U]\), the number of distinct sizes of jobs in this interval is at most

\[
\log_{1+\varepsilon} \frac{U}{L} + 2.
\]

**Proof** At the end of the first rounding step the number of distinct sizes of jobs in this interval is at most \( \log_{1+\varepsilon} \frac{U}{L} + 1 \), and this number may increase by at most 1 due to the second rounding step (in case where we round down a value which is slightly larger than \( U \)).

Next, we summarize the degradation of the approximation ratios due to the two rounding steps.

**Lemma 11** Let \( \varepsilon > 0 \). Assume that there exists an approximation algorithm for instances of the minimization problem resulting from the two rounding steps with an approximation ratio \( 1 + t\varepsilon \) and running time \( T(n, \varepsilon) \) (where \( t \) is a constant), then the approximation ratio of the algorithm (for the original instance of the minimization problem) is \( (1 + t\varepsilon) \cdot (1 + \varepsilon)^{3p} \) and its running time is \( T(n, \varepsilon) + O(n) \).

Assume that there exists an approximation algorithm for instances of the maximization problem resulting from the two rounding steps with an approximation ratio \( 1 + t\varepsilon \) and running time \( T(n, \varepsilon) \) (where \( t \) is a constant), then the approximation ratio of the algorithm (for the original instance of the maximization problem) is \( (1 + t\varepsilon) \cdot \frac{(1+\varepsilon)^{2p}}{1-\varepsilon} \) and its running time is \( T(n, \varepsilon) + O(n) \).

### 3 Approximating the problem with a bounded weight ratio

In this section we consider the following variant of our (maximization or minimization) problem which is called **Bounded Ratio** (BR). The machine set is a subset of machines of consecutive indices from \( I \), and sizes of jobs come from this instance as well.

The input to this problem consists of the following parts:
1. A set of \( \ell \geq 1 \) consecutive machines with speeds \( s_i, s_{i+1}, \ldots, s_{i+\ell-1} \), for which we define

\[
s_{\text{min}} = \min \{ s_i, s_{i+1}, \ldots, s_{i+\ell-1} \} \text{ and } s_{\text{max}} = \max \{ s_i, s_{i+1}, \ldots, s_{i+\ell-1} \}
\]

and assume \( \frac{s_{\text{max}}}{s_{\text{min}}} \leq \beta(\varepsilon) \). Recall that for the minimization problem we assume that the speeds are non-decreasing and thus \( s_{\text{min}} = s_i \) and \( s_{\text{max}} = s_{i+\ell-1} \), and for the maximization problem we assume that the speeds are non-increasing and thus \( s_{\text{min}} = s_{i+\ell-1} \) and \( s_{\text{max}} = s_i \).

2. Two values \( \mathcal{W}_i \) and \( \mathcal{W}_{i+\ell-1} \), such that \( \mu \leq \mathcal{W}_i \leq \mathcal{W}_{i+\ell-1} \), where \( \frac{\mathcal{W}_{i+\ell-1}}{\mathcal{W}_i} \leq \gamma(\varepsilon) \). The values \( \mathcal{W}_i, \mathcal{W}_{i+\ell-1} \) must be (not necessarily positive) integer powers of \( 1 + \varepsilon \). These values are meant to bound the weights of machines, such that they will be integer multiples of \( \mu \) in \( [\mathcal{W}_i, \mathcal{W}_{i+\ell-1}] \) (we will consider approximation algorithms, so we will allow the weights to be slightly smaller or slightly larger, see below).

3. An integer multiple of \( \mu \) called \( A \). For the minimization problem \( A \) is an upper bound on the total size of jobs that the algorithm does not necessarily need to assign to any of these machines, whereas for the maximization problem the value of \( A \) is a lower bound of the total size of jobs that the algorithm should not assign to any of these machines.

4. A set \( L \) of large jobs \( 1, 2, \ldots, n \), each of size at least \( \varepsilon \mathcal{W}_i \).

5. A value \( B \) that corresponds to the total size of existing small jobs that should be treated (if not the entire amount will be assigned, the remainder will be counted towards the total size of unassigned jobs). Small jobs are jobs of size below \( \varepsilon \mathcal{W}_i \), and such jobs can be assigned fractionally. The value \( B \) is an integer multiple of \( \mu \).

As mentioned above, the goal is to schedule the large jobs and the small jobs on the \( \ell \) machines such that the weight of each machine is an integer multiple of \( \mu \) that is at least \( \mathcal{W}_i \) and at most \( \mathcal{W}_{i+\ell-1} \). In particular, machine \( i \) will have a weight of at least \( \mathcal{W}_i \), and machine \( i + \ell - 1 \) will have a weight of at most \( \mathcal{W}_{i+\ell-1} \). We allow an arbitrary subset of the jobs (out of the large jobs and small jobs) to remain unscheduled as long as in the minimization problem the total size of the unscheduled jobs is at most \( A \), and in the maximization problem the total size of the unscheduled jobs must be at least \( A \). We assume that such an assignment of the jobs (that satisfies all properties) is feasible, or else the algorithm returns FALSE. The goal is to minimize or maximize the value \( \sum_{j=i}^{i+\ell-1} C_p^j \) of the schedule, and the input to BR consists of \( i, \ell, \mathcal{W}_i, \mathcal{W}_{i+\ell-1}, A, \) \( L \), and \( B \). Later, we will allow the algorithm (but not the optimal solution) to use machines with weights in the interval \( \left[ \frac{\mathcal{W}_i}{1+3\varepsilon}, \mathcal{W}_{i+\ell-1} \cdot (1 + 3\varepsilon) \right] \).

We will initially analyze solutions where the weights are in \( [\mathcal{W}_i, \mathcal{W}_{i+\ell-1}] \), and later we will allow weights in the slightly extended interval in some cases.

**Remark 12** By Corollary 4, the requirement that \( \frac{s_{\text{max}}}{s_{\text{min}}} \leq \beta(\varepsilon) \) follows from \( \frac{\mathcal{W}_{i+\ell-1}}{\mathcal{W}_i} \leq \gamma(\varepsilon) \).

Next, the number of different sizes of large jobs is at most \( \log_{1+\varepsilon} \mathcal{W}_{i+\ell-1} \mathcal{W}_i + 2 \leq \log_{1+\varepsilon} \gamma(\varepsilon) + 2 \), and thus this number is a function of \( \varepsilon \). Let \( H \) be the set of different
sizes of large jobs, and for each \( h \in H \) we let \( n_h \) be the number of jobs of size \( h \). We define a class of machines to be machines with the same (rounded) speed. Since the speeds are integer powers of \( 1 + \epsilon \) and the ratio between speeds satisfies \( \frac{\max}{\min} \leq \beta(\epsilon) \), we conclude that the number of non-empty machine classes, denoted as \( \tau(\epsilon) \), is at most \( \log_{1+\varepsilon} \beta(\epsilon) + 1 \) which is a function of \( \epsilon \). We denote the non-empty machine classes in our problem by \( M_1, \ldots, M_{\tau(\epsilon)} \). For each machine class \( M_k \), whose machines have a common speed of \( \sigma_k \), we denote the number of machines in \( M_k \) by \( \nu(\sigma_k) = |M_k| \).

We define a configuration \( K \) of a machine as a vector with the following components. The first \(|H|\) components of \( K \) define the number of large jobs of each size that we schedule on a machine with configuration \( K \). Specifically, for each \( h \in H \), \( n(h, K) \) denotes the number of jobs of size \( h \) that are scheduled on a machine with this configuration. Each \( n(h, K) \) is a non-negative integer such that \( n(h, k) \leq \frac{W_{i+\varepsilon}}{\varepsilon W_i} \leq \gamma(\varepsilon) \), that is, it is bounded from above by a function of \( \epsilon \). The next component of \( K \) is an integer power of \( 1 + \epsilon \) in the range \([W_i, W_i+\ell-1]\) denoted as \( w(K) \) (that is, we always have \( w(K) = (1 + \epsilon)^j \) for some integer value \( j \)). The value \( w(K) \) is the approximate total size of the jobs assigned according to this configuration. Since in the instance \( I \) all sizes of jobs are multiples of \( \mu \), and moreover, in actual schedules for \( I \) the weight of a machine must be a multiple of \( \mu \), we will seek for such solutions here as well. Instead of letting the value \( w(K) \) be the weight of a machine, we define rounded versions of \( w(K) \), denoted as \( \tilde{w}(K) \), and the value \( w(K) \) will always be interpreted as \( \tilde{w}(K) \). For the minimization problem, let \( \tilde{w}(K) = \left\lfloor \frac{w(K)}{\mu} \right\rfloor \cdot \mu \). In this case, \( \tilde{w}(K) \) is the maximum total size of jobs in configuration \( K \). For the maximization problem, let \( \tilde{w}(K) = \left\lceil \frac{w(K)}{\mu} \right\rceil \cdot \mu \). In this case, \( \tilde{w}(K) \) is the minimum total size of jobs in configuration \( K \). The number of options for the component \( w(K) \) is at most \( \log_{1+\varepsilon} \frac{W_{i+\ell-1}}{W_i} + 1 \leq \log_{1+\varepsilon} \gamma(\varepsilon) + 1 \), which is a function of \( \epsilon \). The last component is the speed of machines, and we denote this component by \( s(K) \). There are \( \tau(\epsilon) \) options for this last component. In summary, a configuration is a vector that specifies a schedule of one machine. The first \(|H|\) components of this vector are denoted by \( n(h, K) \) for \( h \in H \), and they are integers in \([0, \gamma(\varepsilon)\varepsilon]\), stating how many jobs of each size of \( H \) should be scheduled on this machine. The next component \( w(K) \) is a power of \( 1 + \epsilon \) in \([W_i, W_i+\ell-1]\), and it is interpreted as \( \tilde{w}(K) \), which is the total size of all jobs assigned to a machine that has configuration \( K \), including both large jobs and small jobs. The last component is the speed of machines that can use this configuration. Note that by this definition, not all configurations can actually be used. For a configuration to be valid, the number of jobs of each size cannot exceed the existing number, and the total size of the large jobs specified by the first \(|H|\) components cannot exceed \( \tilde{w}(K) \).

We conclude that the number of different configurations is a function of \( \epsilon \) and we can enumerate all of them in constant time. More accurately, the number of configurations is at most \( \left( \frac{\gamma(\varepsilon)}{\varepsilon} + 1 \right)^{|H|} \cdot \left( \log_{1+\varepsilon} \frac{\gamma(\varepsilon)}{\varepsilon} + 1 \right) \cdot \tau(\epsilon) \leq \left( \frac{\gamma(\varepsilon)}{\varepsilon} + 1 \right)^{\log_{1+\varepsilon} \left( \frac{\gamma(\varepsilon)}{\varepsilon} + 2 \right)} \cdot \left( \log_{1+\varepsilon} \frac{\gamma(\varepsilon)}{\varepsilon} + 1 \right) \cdot \left( \log_{1+\varepsilon} \beta(\varepsilon) + 1 \right) \). Using \( \log_{1+\varepsilon} \epsilon \leq \frac{\epsilon}{\varepsilon} \), we get the following upper bound on the number of configurations: \( \left( \frac{\gamma(\varepsilon)}{\varepsilon} + 1 \right)^{\left( \frac{\gamma(\varepsilon)}{\varepsilon} + 2 \right)} \cdot \left( \frac{\gamma(\varepsilon)}{\varepsilon} + 1 \right) \cdot \left( \beta(\varepsilon) + 1 \right) \). We denote the set of all configurations by \( \mathcal{K} \). We will use the following properties in our
integer program (IP) formulation. A configuration of a machine defines the number of large jobs of each size that are scheduled on such a machine, as well as the maximum or minimum total size (for the minimization problem and the maximization problem, respectively) of small jobs which are scheduled fractionally on such a machine (this is simply the difference between $\hat{w}(K)$ and the total size of the large jobs). The size of a large job is an integer multiple of $\mu$, and as explained above, we will require that the total size of small jobs which are scheduled on such a machine is an integer multiple of $\mu$ as well.

We define IP’s of fixed dimension that will allow us to solve BR (approximately). For each configuration $K \in \mathcal{K}$, there is a decision variable $x_K$ counting the number of machines whose schedule is exactly according to configuration $K$. We let $y_h$ be the number of large jobs of size $h \in \mathcal{H}$ that will remain unscheduled in our solution (as a part of the total budget on the total size of jobs that can be unscheduled).

The following integer program is used for our minimization problem. The motivation for it is explained later (in the proof of the claim that proves its properties and relation to BR). The IP mainly enforces the conditions of BR, where the only difference between a solution to BR and a solution to the IP is that weights of configurations are powers of $1 + \varepsilon$, while weights of machines are multiples of $\mu$, and there can be a difference within a factor of at most $1 + \varepsilon$ between the two values that may affect the cost (since there can be a number of integer multiples of $\mu$ between two powers of $1 + \varepsilon$, we cannot avoid this possible gap).

$$\min \sum_{K \in \mathcal{K}} \left( \frac{w(K)}{s(K)} \right)^p \cdot x_K$$

s.t.

$$\sum_{K \in \mathcal{K} : s(K) = \sigma_k} x_K = \nu(\sigma_k) \quad \forall k = 1, 2, \ldots, \tau(\varepsilon)$$

$$\sum_{K \in \mathcal{K}} n(h, K) \cdot x_K + y_h = n_h \quad \forall h \in \mathcal{H}$$

$$\sum_{h \in \mathcal{H}} h \cdot y_h - \sum_{K \in \mathcal{K}} x_K \cdot \left( \hat{w}(K) - \sum_{h \in \mathcal{H}} h \cdot n(h, K) \right) \leq A - B$$

$$\sum_{h \in \mathcal{H}} h \cdot y_h \leq A$$

$$x_K, y_h \geq 0 \quad \forall K \in \mathcal{K}, \forall h \in \mathcal{H}.$$ (5)

Let $(x^*, y^*)$ be an optimal solution for this IP, and let $X^* = \sum_{K \in \mathcal{K}} \left( \frac{w(K)}{s(K)} \right)^p \cdot x_K^*$ be its objective function value. Let $\text{OPT}_{br}$ denote an optimal solution for this bounded ratio minimization problem as well as the value of its objective function.

**Claim 13** We have $\text{OPT}_{br} \leq X^* \leq (1 + \varepsilon)^p \cdot \text{OPT}_{br}$ for the minimization problem.

**Proof** For the second inequality, we will show that $\text{OPT}_{br}$ induces a feasible solution for the integer program whose cost is at most $(1 + \varepsilon)^p \cdot \text{OPT}_{br}$. For every machine, we define its configuration $K$ as follows. The speed component $s(K)$ is simply the speed.
of this machine. Let $\omega$ denote the weight of the machine, such that $W_i \leq \omega \leq W_i + \ell - 1$. Let $\omega'$ be the smallest integer power of $1 + \varepsilon$ such that $\omega \leq \omega'$. Since $W_{i+\ell-1}$ is an integer power of $1 + \varepsilon$, we have $W_i \leq \omega' \leq W_i + \ell - 1$, and we let $w(\ell) = \omega'$. Every component $n(h, K)$ is exactly the number of jobs of size $h$ assigned to this machine by $\text{OPT}_{br}$. We have $0 \leq n(h, k) \leq \omega \leq W_{i+\ell-1} \leq \nu(\varepsilon, \ell)$. We define a solution $(x', y')$ as follows. For every $K' \in \mathcal{K}$, let $x'_{K'}$ be the number of machines for which the configuration $K'$ was created. Let $y'_{h}$ be the number of unscheduled jobs of size $h$ in $\text{OPT}_{br}$. We show that all constraints are satisfied by this solution. The family of constraints (1) is used to enforce that we use exactly $\nu(\sigma_k)$ machines with speed $\sigma_k$. Since for every machine, its actual speed was used in its configuration, all these constraints are satisfied. The family of constraints (2) is used to enforce that exactly $y'_{h}$ jobs of size $h$ are unscheduled by our solution. Once again, since the values $y'_{h}$ are taken from the schedule $\text{OPT}_{br}$, while the numbers of large jobs in configurations are actual numbers of jobs, these constraints are satisfied too. Constraints (3) and (4) are used to enforce the condition on the total size of unscheduled jobs. Constraint (4) must hold for $(x', y')$ since the total size of the unscheduled large jobs in any solution cannot exceed $A$. To see that constraint (3) is satisfied, consider $\text{OPT}_{br}$, and assume that machine $i$ has the configuration $K$ and weight $\omega$. Since $\omega$ is an integer multiple of $\mu$, we find $0 \leq \tilde{w}(K)$. The total size of large jobs assigned to $i$ in $\text{OPT}_{br}$ is $\sum_{h \in H} h \cdot n(h, K)$, so the total size of small jobs assigned to it is $\omega - \sum_{h \in H} h \cdot n(h, K) \leq \tilde{w}(K) - \sum_{h \in H} h \cdot n(h, K)$. Since the sizes of all large jobs are integer multiple of $\mu$, and small jobs are assigned fractionally, it is indeed possible to reach this bound exactly if there is a sufficient amount of small jobs. We consider two cases. First, assume that $\sum_{K \in \mathcal{K}} x'_{K} \cdot (\tilde{w}(K) - \sum_{h \in H} h \cdot n(h, K)) \geq B$ holds (this case can be interpreted as the case that the gaps left by the large jobs are sufficient to accommodate all small jobs). In this case constraint 3 follows from 4. Otherwise, the total size of unassigned small jobs in $\text{OPT}_{br}$ is at least $B - \sum_{K \in \mathcal{K}} x'_{K} (\tilde{w}(K) - \sum_{h \in H} h \cdot n(h, K)) > 0$, and the total size of all unassigned jobs is at least $\sum_{h \in H} h \cdot y'_{h} + (B - \sum_{K \in \mathcal{K}} x'_{K} (\tilde{w}(K) - \sum_{h \in H} h \cdot n(h, K)))$ and at most $A$, so constraint (3) is satisfied too. Since for every machine with configuration $K$, $w(K)$ is larger than its work by a factor of at most $1 + \varepsilon$, the objective function value for $(x', y')$ is no larger than $(1 + \varepsilon)^P \cdot \text{OPT}_{br}$.

To prove the first inequality, consider $(x^*, y^*)$. For every $K \in \mathcal{K}$ such that $x^*_K > 0$, create a schedule of $x^*_k$ machines, whose speeds are equal to the last component of $K$, where the assignment for each such machine is created as follows. For any $h \in H$, assign $n(h, K)$ jobs of size $h$ to this machine. If $\sum_{h \in H} h \cdot n(h, K) < \tilde{w}(K)$, add small jobs to the machine (scheduled fractionally) until either no small jobs remain or the machine has weight $\tilde{w}(K)$. The small jobs are assigned in a fractional Next Fit manner. This means that unassigned jobs or parts of jobs are assigned to the machine until it reaches the required work (or no small jobs are left). The last assigned job is possibly cut into two parts such that the work becomes exactly $\tilde{w}(K)$. By constraint (1), the number of machines of each speed is sufficient for allocating $x^*_K$ machines to have an assignment according to configuration $K$. By constraint (2), all jobs of size $h$ except for $y'_{h}$ jobs will be scheduled. If all small jobs are assigned, then by constraint (4), the unassigned jobs will have total size of at most $A$. If not all small jobs are assigned, then the total size of scheduled small jobs is $\sum_{K \in \mathcal{K}} (\tilde{w}(K) - \sum_{h \in H} h \cdot n(h, K)) < B$,
and by constraint (3), the total size of all unassigned jobs does not exceed \( A \). Since the total size of jobs assigned to a machine that uses configuration \( K \) does not exceed \( \tilde{w}(K) \leq w(K) \), the cost of the obtained solution does not exceed \( X^* \). \( \Box \)

The following IP is used for solving our maximization problem.

\[
\max \sum_{K \in \mathcal{K}} \left( \frac{w(K)}{s(K)} \right)^p x_K
\]

\[ \text{s.t.} \quad \sum_{K \in \mathcal{K}, s(K) = \sigma_k} x_K = v(\sigma_k) \quad \forall k = 1, 2, \ldots, \tau(\varepsilon) \quad (6) \]

\[
\sum_{K \in \mathcal{K}} n(h, K) \cdot x_K + y_h = n_h \quad \forall h \in H \quad (7)
\]

\[
\sum_{h \in H} h \cdot y_h - \sum_{K \in \mathcal{K}} x_K \cdot \left( \tilde{w}(K) - \sum_{h \in H} h \cdot n(h, K) \right) \geq A - B \quad (8)
\]

\[
\sum_{K \in \mathcal{K}} x_K \cdot \left( \tilde{w}(K) - \sum_{h \in H} h \cdot n(h, K) \right) \leq B \quad (9)
\]

\[
x_K, y_h \geq 0 \quad \forall K \in \mathcal{K}, \forall h \in H. \quad (10)
\]

Let \((x^*, y^*)\) be an optimal solution for the above integer program, and let \( X^* = \sum_{K \in \mathcal{K}} \left( \frac{w(K)}{s(K)} \right)^p x_K^* \) be its objective function value. Denote by \( \text{OPT}_{br} \) the optimal solution for this bounded ratio optimization problem as well as the value of its objective function.

**Claim 14** We have \( X^* \leq \text{OPT}_{br} \leq (1 + \varepsilon)^p \cdot X^* \) for the maximization problem.

**Proof** For the second inequality, we will show that \( \text{OPT}_{br} \) induces a feasible solution for the integer program whose objective function value of at least \( \frac{\text{OPT}_{br}}{(1 + \varepsilon)^p} \). For every machine, we define its configuration \( K \) as follows. The speed component is simply the speed of this machine. Let \( \omega \) denote the weight of the machine, and we have \( \mathcal{W}_i \leq \omega \leq \mathcal{W}_{i+\varepsilon} \). Let \( \omega' \) be the largest integer power of \( 1 + \varepsilon \) such that \( \omega \geq \omega' \). Since \( \mathcal{W}_i \) is an integer power of \( 1 + \varepsilon \), we have \( \mathcal{W}_i \leq \omega' \leq \mathcal{W}_{i+\varepsilon} \), and we let \( w(K) = \omega' \). Every component \( n(h, K) \) is exactly the number of jobs of size \( h \) assigned to this machine by \( \text{OPT}_{br} \). In this case we also have \( 0 \leq n(h, k) \leq \frac{\omega}{\varepsilon \mathcal{W}_i} \leq \frac{\mathcal{W}_{i+\varepsilon}-1}{\varepsilon \mathcal{W}_i} \leq \frac{\gamma(\varepsilon)}{\varepsilon} \).

We define a solution \((x', y')\) as follows. For every \( K' \in \mathcal{K} \), let \( x'_{K'} \) be the number of machines for which the configuration \( K' \) was created. Let \( y'_h \) be the number of unscheduled jobs of size \( h \) in \( \text{OPT}_{br} \). We show that all constraints are satisfied by this solution. The family of constraints (6) is used to enforce that we use exactly \( v(\sigma_k) \) machines with speed \( \sigma_k \). Since for every machine, its actual speed was used in its configuration, all these constraints are satisfied. The family of constraints (7) is used to enforce that exactly \( y'_h \) jobs of size \( h \) are unscheduled by our solution. Once again, since the values \( y'_h \) are taken from the schedule \( \text{OPT}_{br} \), while the numbers of large jobs in configurations are actual numbers of jobs, so these constraints are satisfied too. Constraints (8) and (9) are used to enforce the condition on the total size of unscheduled jobs assigned to a machine that uses configuration \( K \) does not exceed \( \tilde{w}(K) \leq w(K) \), the cost of the obtained solution does not exceed \( X^* \). \( \Box \)
jobs. Since $\omega$ is an integer multiple of $\mu$, we find $\omega \geq \tilde{\omega}(K)$. The total size of small jobs of a machine of weight $\omega$ that has the configuration $K$ is $\omega - \sum_{h \in H} h \cdot n(h, K) \geq \tilde{\omega}(K) - \sum_{h \in H} h \cdot n(h, K)$. Thus, the total size $B$ of existing small jobs is at least $\sum_{K \in \mathcal{K}} x'_K (\tilde{\omega}(K) - \sum_{h \in H} h \cdot n(h, K))$, and constraint (9) holds for $(x', y')$. Next, we show that constraint (8) is satisfied. It is definitely satisfied if $\sum_{h \in H} h \cdot y'_K \geq A$. If this is not the case, then the total size of unscheduled small jobs is at least $A - \sum_{h \in H} h \cdot y'_K$, and it is at most $B - (\sum_{K \in \mathcal{K}} x'_K (\tilde{\omega}(K) - \sum_{h \in H} h \cdot n(h, K)))$, proving constraint (8). Since for every machine with configuration $K$, $w(K)$ is smaller than its work by a factor of at most $1 + \epsilon$, the objective function value for $(x', y')$ is at least $\frac{\text{OPT}_{br}}{(1 + \epsilon)^p}$.

To prove the first inequality, consider $(x^*, y^*)$. For every $K \in \mathcal{K}$ such that $x^*_K > 0$, create a schedule of $x^*_K$ machines, whose speeds are equal to the last component of $K$, where the assignment for each such machine is created as follows. For any $h \in H$, assign $n(h, K)$ jobs of size $h$ to this machine. If $\sum_{h \in H} h \cdot n(h, K) < \tilde{\omega}(K)$, then add small jobs to the machine (scheduled fractionally) until the machine has weight $\tilde{\omega}(K)$. The small jobs are again assigned in a fractional Next Fit manner, and in this case all gaps are filled so that the resulting work becomes exactly $\tilde{\omega}(K)$. By constraint (9), the total size of existing small jobs is sufficient for this process. By constraint (6), the number of machines of each speed is sufficient for allocating $x^*_K$ machines to have an assignment according to configuration $K$. By constraint (7), all jobs of size $h$ except for $y^*_K$ jobs will be scheduled. If all small jobs are assigned, then by constraint (8), the unassigned jobs will have total size of at least $A$. Since the total size of jobs assigned to a machine that uses configuration $K$ is exactly $\tilde{\omega}(K) \geq w(K)$, the value of the obtained solution is at least $X^*$.

Claim 15 The integer programs can be solved in strongly polynomial time. More precisely, the running time of solving the integer program is at most

$$O \left( \left( \frac{\gamma(\epsilon)}{\epsilon} + 1 \right)^{\frac{p(\epsilon) + 2}{\epsilon^2}} \left( \frac{\gamma(\epsilon)}{\epsilon^2} + 1 \right) \cdot \left( \frac{\beta(\epsilon)}{\epsilon} + 1 \right) \right)^{\frac{p(\epsilon) + 2}{\epsilon^2}} \cdot \tau(\epsilon) \cdot \log n \right).$$

Proof First, the construction of the integer programs takes polynomial time since the set of all configurations can be enumerated in polynomial time (using the fact that $\mathcal{K}$ has at most a constant number of configurations which is bounded from above by a function of $\epsilon$). Next, we observe that the dimension of each of these programs (the number of variables) is $|\mathcal{K}| + |H| = O \left( \frac{\gamma(\epsilon)}{\epsilon} + 1 \right)^{\frac{p(\epsilon) + 2}{\epsilon^2}} \cdot \left( \frac{\gamma(\epsilon)}{\epsilon^2} + 1 \right) \cdot \left( \frac{\beta(\epsilon)}{\epsilon} + 1 \right)$. Thus the integer program has a fixed dimension, and we can use the polynomial time...
algorithms for solving such a problem. The number of constraints (that are not non-negative constraints) is \( \tau(\varepsilon) + |H| + 2 \) which is again bounded from above by a function of \( \varepsilon \). Therefore, we can use Lenstra’s algorithm [29] or one of its improvements (see e.g. [28]) that gave polynomial time algorithms for solving (exactly) the integer programs (recall that the time complexity of solving an integer program of dimension \( d \) is \( f(d) \cdot \text{poly} \) where \( f \) is an exponential function of the dimension, and \( \text{poly} \) is a polynomial in the binary encoding of the program). Here, we compute the running time resulting from using Kannan’s algorithm [28]. The running time for solving integer program in dimension \( d \) whose binary encoding length is \( L \) is \( O(d^{9d} L \log L) \).

To obtain a strongly polynomial time we use the following observations. First, the coefficients in the objective function are integer powers of \( 1 + \varepsilon \) and can be scaled to be at most \( (\gamma(\varepsilon) + \varepsilon) \cdot (\gamma(\varepsilon) + 1) \). Next, we scale constraints (3), (4), (8), and (9) by dividing the constraints by the factor \( \mu \). In the resulting constraint matrices and right hand sides, all the coefficients are strongly polynomial (i.e., do not depend on the magnitude of the numbers in the instance), and are at most \( n^2 m^{1/p} \varepsilon^{1/p + 2} \). Thus, \( L = O \left( (\log n + \log m + \log \frac{1}{\varepsilon}) \cdot (|K| + |H|) \cdot (\tau(\varepsilon) + |H|) \right) \). Thus, the running time of solving the integer program follows.

We find that given an optimal solution for one of the IP’s, it is possible to convert it into a schedule in linear time. Thus, we conclude the following result.

**Proposition 16** Problem BR has an EPTAS. That is, there exists an algorithm with approximation ratio \((1 + \varepsilon)^p\) and running time

\[
O \left( \left( \frac{\gamma(\varepsilon)}{\varepsilon} + 1 \right) \left( \frac{\gamma(\varepsilon)}{\varepsilon^2} + 2 \right) \cdot \left( \frac{\gamma(\varepsilon)}{\varepsilon^2} + 1 \right) \right)
\]

\[
\cdot \left( \frac{\beta(\varepsilon)}{\varepsilon} + 1 \right) \cdot \tau(\varepsilon) \cdot \log n
\]

We next consider a variant of BR in which the small jobs that are scheduled on one of the \( \ell \) machines need to be scheduled integrally. We call the resulting problem \textsc{Integer Bounded Ratio} (IBR). In order to obtain the EPTAS for IBR, we note that in our algorithm for BR, each machine receives at most two small jobs fractionally. For the maximization problem of IBR we simply remove the fractional parts (and assign them arbitrarily). This decreases the work of each machine by at most \( 2\varepsilon W_i \), and thus the completion time of each machine is decreased by a multiplicative factor of at most \( 1 - 2\varepsilon \). This gives an EPTAS for the maximization problem of IBR. For the minimization problem we assign (integrally) each small job to the first machine that gets a fraction of the job in the solution to BR. This may increase...
the work of a machine by at most $\epsilon N_i$. Therefore, the completion time of each machine increases by a multiplicative factor of at most $1 + \epsilon$ (it may decrease for some machines as well). Hence, in this case the total cost of the resulting solution to IBR is at most $(1 + \epsilon)^p$ times the cost of the solution to BR. Thus, this gives an EPTAS for the minimization problem of IBR as well, and the following result is established (the running time does not increase asymptotically if IBR is solved rather than BR).

**Theorem 17** Problem IBR has an EPTAS. The approximation ratio of this scheme is $(1 + \epsilon)^2p$ for the minimization problem, and $\frac{(1 + \epsilon)p}{(1 - 2\epsilon)p}$ for the maximization problem. The running time of the scheme is the same as the scheme for problem BR.

4 Applying the shifting technique

In this section we use the shifting technique of Hochbaum and Maas [22]. Our goal is to restrict ourselves to looking for schedules where the weights of machines cannot come from a specified set of intervals. Our algorithm will choose the best outcome among a constant number of iterations. In each such iteration, we will use a different set of illegal intervals. Let $p_{\text{min}} = \min_j p_j$. A set of illegal intervals is a set $S = \{(a_0, b_0), (a_1, b_1), \ldots, (a_r, b_r)\}$ where $b_\ell < a_{\ell+1}$ for all $0 \leq \ell \leq r - 1$, and additionally, $a_0 \geq p_{\text{min}}$ and $a_r \leq n p_{\text{max}}$. We also let $b_{-1} = p_{\text{min}}$ and $a_{r+1} = n p_{\text{max}}$.

Let $\rho(\epsilon) = \frac{1}{\epsilon^2(\rho(\epsilon))^p} \geq \frac{1}{\epsilon^p}$ (where equality holds for $p < 1$), then for $\eta = 0, 1, 2, \ldots, 2\rho(\epsilon) - 1$, in iteration $\eta$ we will use the following set of illegal intervals:

$$
(a_{\ell}^{\eta}, b_{\ell}^{\eta}) = \left(p_{\text{min}} \cdot \left(\frac{1}{\epsilon}\right)^{\eta + 2\ell\rho(\epsilon)}, p_{\text{min}} \cdot \left(\frac{1}{\epsilon}\right)^{\eta + 2\ell\rho(\epsilon) + 1}\right),
$$

for the non-negative values of $\ell$ such that $a_{\ell}^{\eta} \leq n p_{\text{max}}$. In our algorithm we will apply the EPTAS for IBR (multiple times) on every valid interval that results from removing invalid intervals in some iteration. Every such interval is contained in $[p_{\text{min}}, n p_{\text{max}}]$ and it starts at a point of the form $p_{\text{min}} \cdot \left(\frac{1}{\epsilon}\right)^k$ for some integer $0 \leq k \leq \log_2 \frac{np_{\text{max}}}{p_{\text{min}}}$. Thus, the number of intervals that will be valid for some iteration is at most

$$
\log_2 \frac{np_{\text{max}}}{p_{\text{min}}} + 1 \leq \log_2 \frac{np_{\text{max}}}{\mu} + 1 \leq \log_2 \left(\frac{n^2 \cdot m^{1/p}}{\epsilon^{1/p + 2}}\right) + 1,
$$

and this is a polynomial in $n, m$ and $\frac{1}{\epsilon}$. We denote by $S_\eta$ the set of illegal intervals of iteration $\eta$. Let $\lambda(\epsilon) = 2\rho(\epsilon) - 1$ (and recall that $\gamma(\epsilon) = \frac{1}{\epsilon^{(1/\lambda(\epsilon))}} = \frac{1}{\epsilon^{\ln(\epsilon)/\epsilon - 1}}$).

For a given $S_\eta$, we say that two values $W, W'$ that do not belong to intervals of $S_\eta$ such that $p_{\text{min}} \leq W < W' \leq n p_{\text{max}}$, are separated by an interval of $S_\eta$ if there exists a value $\ell$ such that $W \leq a_{\ell}^{\eta}$ and $W' \geq b_{\ell}^{\eta}$, and otherwise they are not separated.

**Observation 18** If $W, W'$ are separated by an interval from $S_\eta$, then $\frac{W'}{W} \geq \frac{1}{\epsilon}$, and otherwise $\frac{W'}{W} \leq \gamma(\epsilon)$.

**Proof** The first part holds since $\frac{b_{\ell}^{\eta}}{a_{\ell}^{\eta}} = \frac{1}{\epsilon}$ for any $\ell$. The second part holds since for any $\ell$, $\frac{a_{\ell}^{\eta}}{b_{\ell+1}^{\eta}} \leq \gamma(\epsilon)$.

© Springer
Let $$\text{OPT}_\eta(I)$$ denote the cost of an optimal solution where no machine of positive weight has a weight in an interval of $$S_\eta$$. The correctness of removing the intervals of $$S_\eta$$ for some $$\eta$$ is implied using the next theorem.

**Theorem 19** For the minimization problem, there exists a value of $$\eta$$ such that $$\text{OPT}_\eta(I) \leq (1 + \varepsilon)\text{OPT}(I)$$. For the maximization problem, there exists a value of $$\eta$$ such that $$\text{OPT}_\eta(I) \geq (1 - \varepsilon^3)\text{OPT}(I)$$.

**Proof** Fix an optimal solution $$\text{OPT}$$ for $$I$$, where the weight of machine $$i$$ is $$W_i$$.

Consider the maximization problem first. Let $$0 \leq \eta' \leq \rho(\varepsilon) - 1$$, and let $$M_{\eta'}$$ the subset of machines whose weight is in $$S_{\eta'} \cup S_{\eta'} + \rho(\varepsilon)$$. Let the part of the profit for these machines be $$\Phi_\eta'$$ (so $$\text{OPT}(I) = \sum_{\eta' = 0}^{\rho(\varepsilon) - 1} \Phi_\eta'$$). By the pigeonhole principle, there exists a value of $$\eta'$$, denoted by $$\tilde{\eta}$$ for which $$\Phi_{\tilde{\eta}} \leq \frac{\text{OPT}(I)}{\rho(\varepsilon)} = \varepsilon^3\text{OPT}(I)$$. We create an alternative solution $$\text{SOL}$$ from $$\text{OPT}$$. In this solution, all machines except for the machines of $$M_{\eta'}$$ and machine $$M_m$$ have the same jobs, and all jobs that were previously assigned to machines of $$M_{\eta'}$$ are assigned to $$M_m$$ (no matter whether $$M_m$$ belongs to $$M_{\eta'}$$ or not). As a result, the profit is reduced by at most $$\Phi_{\tilde{\eta}}$$ (removing the jobs from their positions reduces the profit by this amount, while assigning additional jobs to $$M_m$$ cannot reduce the profit). The only machine of $$\text{SOL}$$ that may have weight in an interval of $$S_{\tilde{\eta}} \cup S_{\tilde{\eta} + \rho(\varepsilon)}$$ is $$M_m$$. If its weight is in $$S_{\tilde{\eta}}$$, then no machine has weight in $$S_{\tilde{\eta} + \rho(\varepsilon)}$$, and if its weight is in $$S_{\tilde{\eta} + \rho(\varepsilon)}$$, then no machine has weight in $$S_{\tilde{\eta}}$$. Thus $$\text{OPT}_{\tilde{\eta}}(I) \geq (1 - \varepsilon^3)\text{OPT}(I)$$ or $$\text{OPT}_{\tilde{\eta} + \rho(\varepsilon)}(I) \geq (1 - \varepsilon^3)\text{OPT}(I)$$ (or both) must hold. Thus, the claim for the maximization problem follows.

For the minimization problem, we define a function $$g_{\eta'} : [0, \sum_{j=1}^{n} p_j] \rightarrow \mathbb{R}^+_0$$ as follows ($$\mathbb{R}^+_0$$ denotes the non-negative reals). We let $$g_{\eta'}(x) = x$$ unless $$x$$ belongs to an interval of $$S_{\eta'} \cup S_{\eta'} + \rho(\varepsilon)$$, $$(a_i^\eta, b_i^\eta)$$ (where $$\eta = \eta'$$ or $$\eta = \eta' + \rho(\varepsilon)$$), in which case $$g_{\eta'}(x) = 2 \cdot b_i^\eta$$. For a solution $$\text{SOL}'$$ for $$I$$, the cost $$C_{\eta'}(\text{SOL}'(I))$$ is the sum of $$p$$ powers not of the machine completion times, but of the values resulting from applying $$g_{\eta'}$$ on those values (before taking the $$p$$ powers). Obviously, $$\text{SOL}'(I) \leq \text{SOL}''(\text{SOL}'(I))$$. On the other hand, once again we can choose $$0 \leq \tilde{\eta} \leq \rho(\varepsilon) - 1$$ such that $$\Phi_{\tilde{\eta}} \leq \frac{\text{OPT}(I)}{\rho(\varepsilon)}$$ (here $$\Phi_{\tilde{\eta}}$$ denotes cost rather than profit). We modify $$\text{OPT}$$ as follows. If no machine has weight in an interval of $$S_{\eta'} \cup S_{\eta'} + \rho(\varepsilon)$$, then $$\text{OPT}_{\tilde{\eta}}(I) = \text{OPT}(I)$$, and we are done. Otherwise, create a solution $$\text{SOL}$$ as follows. As long as there are at least two machines of weights in $$S_{\eta'} \cup S_{\eta'} + \rho(\varepsilon)$$, choose two such machines, and move all jobs from one of them to the other (if the weights are different, then the jobs are moved from the machine of smaller weight to the machine of larger weight). We show that in each such step, if the resulting solution is denoted by $$\text{SOL}$$, and the previous one was $$\hat{\text{SOL}}$$ then $$G_{\eta'}(\hat{\text{SOL}}(I)) \leq G_{\eta'}(\text{SOL}(I))$$. It is sufficient to consider a single step, and the machine that received additional jobs. Assume that a machine had weight in $$(a_i^\eta, b_i^\eta)$$ (where $$\eta = \eta'$$ or $$\eta = \eta' + \rho(\varepsilon)$$). Its new weight is at most twice its previous weight. If its weight remains in this interval, then $$G_{\eta'}(\text{SOL}(I)) \leq G_{\eta'}(\hat{\text{SOL}}(I)), $$ since applying $$g_{\eta'}$$ on the weight has the same result, and the other machine of modified weight has weight zero in $$\text{SOL}$$. Otherwise, since its new weight is at most twice its original weight (which was below $$b_i^\eta$$), if the new weight is at least $$b_i^\eta$$, then it is at most $$2b_i^\eta < \frac{b_i^\eta}{\varepsilon}$$ (by the definition
of $S_\eta$, since $\varepsilon < \frac{1}{2}$). i.e., it is not in an interval of $S_{\eta'} \cup S_{\eta'} + \rho(\varepsilon)$. Thus, applying $g_{\eta'}$ on the new weight results in a value that is smaller or equal to the value resulting from applying it on the old weight, and we find $G_{\eta'}(\hat{\text{SOL}}(I)) \leq G_{\eta'}(\hat{\text{SOL}}(I))$. The process ends when there is at most one machine of weight in $S_{\eta'} \cup S_{\eta'} + \rho(\varepsilon)$. The achieved solution $\text{SOL}$ has $G_{\eta'}(\text{SOL}(I)) \leq G_{\eta'}(\text{OPT}(I))$. Similar to the maximization problem, since there is at most one machine whose weight is in an interval of $S_{\eta'} \cup S_{\eta'} + \rho(\varepsilon)$, it either has no machines with weights in intervals of $S_{\eta'} + \rho(\varepsilon)$ or no machines with weights in intervals of $S_{\eta'}$ (or both). Since $G_{\eta'}(\text{SOL}(I)) \geq \text{SOL}(I)$, we have shown $\text{SOL}(I) \leq G_{\eta'}(\text{OPT}(I))$, so it is left to show $G_{\eta'}(\text{OPT}(I)) \leq (1 + \varepsilon)\text{OPT}(I)$, or alternatively, to bound the amount $G_{\eta'}(\text{OPT}(I)) - \text{OPT}(I)$ and show that it is at most $\varepsilon \text{OPT}(I)$. For this, we will find an upper bound on the cost of machines of weight in $S_{\eta'} \cup S_{\eta'} + \rho(\varepsilon)$ in $\text{OPT}$. For every such machine, applying $g$ on its weight increases it by at most a multiplicative factor of $\frac{2}{\varepsilon}$. Thus, the new cost for these machines is at most their cost without applying $g_{\eta'}$ times $(\frac{2}{\varepsilon})^p$. We get $G_{\eta'}(\text{OPT}(I)) - \text{OPT}(I) \leq \frac{\text{OPT}(I)}{\rho(\varepsilon)} \cdot (\frac{2}{\varepsilon})^p \leq \frac{\text{OPT}(I)}{\rho(\varepsilon)} \cdot (\frac{1}{\varepsilon})^{2p}$ (since $\varepsilon \leq \frac{1}{2}$). By definition $\rho(\varepsilon) \geq \frac{1}{\varepsilon^{p+1}}$, and therefore $G_{\eta'}(\text{OPT}(I)) - \text{OPT}(I) \leq \varepsilon \text{OPT}(I)$.  

\section{Dynamic programming to approximate $\text{OPT}_\eta$}

Given a fixed value of $\eta$, the set of illegal weights $S_\eta$ leaves a set of valid intervals whose possible weights we denote by $\Omega = \{(\omega_0, \omega_1), (\omega_2, \omega_3), \ldots, (\omega_{r-1}, \omega_r)\}$, where the sequence $\omega_i$ is monotone increasing, $\omega_0 \geq p_{\min}$, and $\omega_{r} \leq n p_{\max}$ (and the number of such intervals for all values of $\eta$ is at most $\log \frac{B^2 \cdot m^{1/p}}{\varepsilon^{p+1}} + 1$). We require that for any job $j$, $\omega_j \leq \omega_{r'}$ (otherwise there is no valid solution). Recall that some machines may get a zero weight in an optimal solution, but we can guess their number and remove the set of these machines (of lowest indices) from the instance. Hence, without loss of generality it suffices to consider solutions that assign at least one job to each machine, and therefore the minimum weight of a machine is at least $\omega_0$.

By Observation 18, we conclude that for any value of $\xi$, $\frac{\omega_{2\xi}}{\omega_{2\xi - 1}} \geq \frac{1}{\varepsilon}$ and $\frac{\omega_{2\xi + 1}}{\omega_{2\xi}} \leq \gamma(\varepsilon)$. We define a linear order on the intervals of $\Omega$ saying that an interval $[\omega_\xi, \omega_{\xi+1}]$ is smaller than $[\omega_{\eta'}, \omega_{\eta'+1}]$ if $\xi < \xi'$, and an interval $[\omega_\xi, \omega_{\xi+1}]$ is at most an interval $[\omega_{\eta'2}, \omega_{\eta'+2}]$ if $\xi \leq \xi'$. We next describe an allocation of jobs to intervals of $\Omega$ according to their sizes. A job $j$ of size $p_j$ is associated with an interval $[\omega_{\xi-1}, \omega_{\xi}]$ if $p_j \leq \omega_{\xi}$ and $p_j > \omega_{\xi-2}$ where we use the convention $\omega_{-1} = 0$. We define an assignment of intervals of (consecutive) machines to intervals of $\Omega$ in the following sense. An interval of machines $[i, i']$ with parameters $A, B$ is assigned to an interval $[\omega_{\xi}, \omega_{\xi+1}] \in \Omega$ if the following four conditions hold:

1. The weight of each machine $\tilde{\xi} \in [i, i' - 1]$ is in the interval $[\omega_{\xi}, \omega_{\xi+1}]$.
2. No other machine has weight in this interval.
3. The total size of jobs associated with smaller intervals and are scheduled on a machine whose index is at least $i$ is exactly $B$.
4. Similarly, the total size of jobs associated with intervals that are at most $[\omega_{\xi}, \omega_{\xi+1}]$, and are scheduled on machines in $[i', m]$ is exactly $A$. Here $A$ and $B$ are integer multiples of $\mu$. We say that $A$ and $B$ are the parameters of the interval of machines $[i, i']$.  

\[ \odot \] Springer
Claim 20 The number of possibilities of a machine interval, a pair of parameters \( A, B \), and a weight interval is polynomial in the input size, and it is at most
\[
(\log \frac{n^{2+1/p}}{\varepsilon^{1/p+2}} + 1) \cdot \frac{n^{b+2/p}}{\varepsilon^{2/p+n}}.
\]

Proof As explained earlier, the number of different valid weight intervals (for all values of \( \eta \)) is at most \( \log \frac{n^{2+1/p}}{\varepsilon^{1/p+2}} + 1 \). The number of pairs of machines \( i \) and \( i' \) is at most \( m^2 \leq n^2 \). A subset of jobs has a total size that is an integer multiple of \( \mu \), and this total size does not exceed \( n \cdot p_{\text{max}} \). Thus, the number of possibilities for the value \( A \) (and similarly for \( B \)) is at most \( \frac{n^2 \cdot p_{\text{max}}}{\mu} \leq \frac{n^2 \cdot m^{1/p}}{\varepsilon^{1/p+2}} \), proving the claim. \( \square \)

Note that the number of possibilities in the claim corresponds to a total number of options, and not only for one specific value of \( \eta \), for which the specific graph is constructed, that is, this is the number of possibilities for all constructed graphs. The EPTAS for IBR will be applied for every option, and the outputs will be used for building the graphs as explained in this section. In the graph construction we assume that all \( m \) machines receive non-zero weight. Since we enumerate all options of the number of machines that do not receive any jobs, the construction below is applied not only for \( m \) but for any \( m' \leq m \). However, this does not affect the running times since asymptotically the running time is determined by applying the EPTAS for IBR for all cases.

Note that given an interval of machines \([i, i']\), and two values \( A, B \), we get an instance of the IBR problem for which we presented an EPTAS (this IBR instance has an empty set of machines if \( i = i' \) and otherwise at least one machine). Thus our scheme applies this EPTAS for each possibility of a machine interval \([i, i']\) such that \( i \leq i' \) corresponding to weight interval \([\omega_\xi, \omega_{\xi+1}]\) with values \( A, B \). We denote the solution returned by the EPTAS for the IBR instance as well as its objective function value by \( IBR_{\text{eptas}}(i, i' - 1, \xi, A, B) \). If \( i = i' \) and \( A = B \), then \( IBR_{\text{eptas}}(i, i' - 1, \xi, A, B) = 0 \), and if \( i = i' \) and \( A \neq B \) then \( IBR_{\text{eptas}}(i, i' - 1, \xi, A, B) \) returns \text{FALSE}. If the returned output is \text{FALSE}, then the value is \( \infty \) for the minimization problem and \( -\infty \) for the maximization problem.

To find the approximate solution for the minimization problem, we find a shortest (minimum cost) path in the layered graph \( G \) defined in what follows, and to find the approximated solution for the maximization problem we find a longest (maximum cost) path in this graph \( G \) (since \( G \) is a layered graph, it is acyclic and hence both the shortest path problem and the longest path problem are solvable in linear time as a function of the number of vertices and arcs). We have a layer for each value of \( \xi \) such that \([\omega_\xi, \omega_{\xi+1}] \in \Omega \) (that is, we will have a layer for every even value of \( \xi \)). Each such layer corresponding to \( \xi \) has some vertices for each machine \( i \); this corresponds to the case that the machines \( 1, \ldots, i - 1 \) were already dealt with, and therefore we use one additional dummy machine of index \( m + 1 \). For every machine, there is a vertex for each possibility for the value of \( A \) (i.e., for each integer multiple of \( \mu \)). Given a vertex \((i, b)\) in layer \( \xi \) and a vertex \((i', a)\) in layer \( \xi + 2 \), there is an arc from the former vertex to the latter vertex if \( i \leq i' \), and the cost associated with such an arc is \( IBR_{\text{eptas}}(i, i' - 1, \xi, a, b) \). The construction of \( G \) takes polynomial time, and it has a polynomial size.
We next find a shortest or longest path in $G$ from the vertex $(1, 0)$ of the layer with index 0 to the vertex $(m + 1, 0)$ in the last layer. We schedule the jobs according to the path which we found. That is, if the path uses the arc from $(i, b)$ of layer $\xi$ to $(i', a)$ of layer $\xi + 2$ where $i \leq i'$, then we schedule large and small jobs as defined by the solution $I\!B\!R_{\text{opt}}(i, i' - 1, \xi, a, b)$. The total size of jobs that are associated with the interval $[\omega_{\xi}, \omega_{\xi + 1}]$ or smaller intervals, and are not scheduled on machines of index at most $i' - 1$ is indeed at most $a$ for the minimization problem and at least $a$ for the maximization problem. Note that every job is scheduled by one of the solutions corresponding to this path, and hence we obtained a feasible solution whose objective function value is the total cost of the arcs in the path.

We next note that $\text{OPT}_{\eta}$ also corresponds to a path in $G$ in the following sense. If $\text{OPT}_{\eta}$ uses machines $i, i + 1, \ldots, i' - 1$ with weight in the interval $[\omega_{\xi}, \omega_{\xi + 1}]$, and assume that the total size of the jobs allocated to smaller intervals and are not scheduled by $\text{OPT}_{\eta}$ to machines with index at most $i - 1$ is $b$, and the total size of the jobs allocated to intervals at most this interval and are not scheduled by $\text{OPT}_{\eta}$ to machines with index at most $i' - 1$ is $a$, then we say that the arc from $(i, b)$ of layer $\xi$ to $(i', a)$ of layer $\xi + 2$ belongs to the path associated with $\text{OPT}_{\eta}$. Since $\text{OPT}_{\eta}$ is a feasible solution, this set of arcs can be augmented to form a path by adding zero cost arcs (from $(i, b)$ of layer $\xi$ to $(i', b)$ of layer $\xi + 2$). Moreover, the value $\text{OPT}_{\eta}(I)$ can be also partitioned into its arcs, by giving the cost (according to $\text{OPT}_{\eta}$) of the machines $i, i + 1, \ldots, i' - 1$ to the arc from $(i, b)$ of layer $\xi$ to $(i', a)$ of layer $\xi + 2$. By Theorem 17, we conclude that the cost given to such an arc in this way is within multiplicative factors of $(1 + \varepsilon)^2p$ and $(1 + \varepsilon)^p$ of the cost of the arc in the graph, for the minimization problem and the maximization problem, respectively. Hence, the cost of this path in $G$ is within a similar multiplicative factor of the objective function value of $\text{OPT}_{\eta}$. Since we use an optimal path, we are not worse off than this path of $\text{OPT}_{\eta}$, and thus we established the following result.

**Theorem 21** Both the problem of minimizing $\sum_{i=1}^{m} C_i^p$ and the problem of maximizing $\sum_{i=1}^{m} C_i^p$ for real finite values of $p$ have efficient polynomial time approximation schemes.

We next summarize the approximation ratio and the running time of the schemes. First consider the approximation ratio for the minimization problem, and observe that by Lemma 11, Theorem 19, and the above discussion, the approximation ratio is $(1 + \varepsilon)^3p \cdot (1 + \varepsilon)^2p \cdot (1 + \varepsilon) = (1 + \varepsilon)^{5p+1}$. Next, we consider the approximation ratio for the maximization problem and conclude that it is $\frac{(1+\varepsilon)^{2p}}{1-\varepsilon} \cdot \frac{(1+\varepsilon)^p}{1-2\varepsilon} \cdot \frac{1}{1-\varepsilon^3} = \frac{(1+\varepsilon)^{3p}}{(1-\varepsilon)(1-\varepsilon^3)(1-2\varepsilon)^p}$.

The running time of the two schemes is $O\left(\frac{m + n \log n \cdot \left(\gamma^{\left(\frac{\gamma(e)}{e}\right)} + 1\right) \left(\frac{\gamma^{\left(\frac{\gamma(e)}{e}\right)} + 1}{\frac{\gamma(e)}{e} + 1}\right) \left(\frac{\gamma^{\left(\frac{\gamma(e)}{e}\right)} + 1}{\frac{\gamma(e)}{e} + 1}\right) \tau(\varepsilon) \cdot \log \left(\frac{e^{\gamma(\varepsilon)}}{\varepsilon^{1/p + 2}}\right) + 1\right) \cdot \frac{N^{6+2/p}}{\varepsilon^{2/p+2}}$, where $\gamma(e) = \frac{1}{\varepsilon^{\gamma(\varepsilon)}}, \beta(\varepsilon) = \frac{2}{\varepsilon^{\beta(\varepsilon)}}$ for the minimization problem, and
\[ \beta(\varepsilon) = \left( \frac{1}{(1+\varepsilon^2(\lambda(\varepsilon) + 1))^{1/p}} \right)^{1/p}, \] where \( \lambda(\varepsilon) = \frac{2}{2^{1/p}+1} - 1 \). Note that this time complexity is the product of a function of \( \varepsilon \) and a polynomial in \( n \) and \( m \). In order to obtain the running time for obtaining a \( 1 + \varepsilon \) approximation, one has to first scale \( \varepsilon \) by a suitable constant so that the approximation ratio will be \( 1 + \varepsilon \), and obtain the resulting running time. The \( O(m) \) additive term appears since before applying the algorithm, if \( n < m \), we find the fastest \( n \) machines.

References

1. Alon, N., Azar, Y., Woeginger, G.J., Yadid, T.: Approximation schemes for scheduling. In: Proceedings of the 8th Symposium on Discrete Algorithms (SODA), pp. 493–500. ACM/SIAM (1997)
2. Alon, N., Azar, Y., Woeginger, G.J., Yadid, T.: Approximation schemes for scheduling on parallel machines. J. Sched. 1(1), 55–66 (1998)
3. Avidor, A., Azar, Y., Sgall, J.: Ancient and new algorithms for load balancing in the \( L_p \) norm. Algorithmica 29(3), 422–441 (2001)
4. Awerbuch, B., Azar, Y., Grove, E.F., Kao, M.-Y., Krishnan, P., Vitter, J.S.: Load balancing in the \( l_\infty \) norm. In: Proceedings of the 36th Symposium Foundations of Computer Science (FOCS), pp. 383–391. IEEE (1995)
5. Azar, Y., Epstein, L.: Approximation schemes for covering and scheduling on related machines. In: Proceedings of the 1st International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX), pp. 39–47 (1998)
6. Azar, Y., Epstein, L., Richter, Y., Woeginger, G.J.: All-norm approximation algorithms. J. Algorithms 52(2), 120–133 (2004)
7. Bansal, N., Pruhs, K.R.: Server scheduling to balance priorities, fairness, and average quality of service. SIAM J. Comput. 39(7), 3311–3335 (2010)
8. Bansal, N., Sviridenko, M.: The Santa Claus problem. In: Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC), pp. 31–40. ACM (2006)
9. Caragiannis, I.: Better bounds for online load balancing on unrelated machines. In: Proceedings of the 19th Symposium on Discrete Algorithms (SODA), pp. 972–981. ACM/SIAM (2008)
10. Cesati, M., Trevisan, L.: On the efficiency of polynomial time approximation schemes. Inf. Process. Lett. 64(4), 165–171 (1997)
11. Chandra, A.K., Wong, C.K.: Worst-case analysis of a placement algorithm related to storage allocation. SIAM J. Comput. 4(3), 249–263 (1975)
12. Cody, R.A., Coffman, E.G. Jr.: Record allocation for minimizing expected retrieval costs on drum-like storage devices. J. ACM 23(1), 103–115 (1976)
13. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Berlin (1999)
14. Efremidis, P., Spirakis, P.G.: Approximation schemes for scheduling and covering on unrelated machines. Theor. Comput. Sci. 359(1–3), 400–417 (2006)
15. Epstein, L., Sgall, J.: Approximation schemes for scheduling on uniformly related and identical parallel machines. Algorithmica 39(1), 43–57 (2004)
16. Epstein, L., van Stee, R.: Maximizing the minimum load for selfish agents. Theor. Comput. Sci. 411(1), 44–57 (2010)
17. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Berlin (2006)
18. Friesen, D.K., Deuermeyer, B.L.: Analysis of greedy solutions for a replacement part sequencing problem. Math. Oper. Res. 6(1), 74–87 (1981)
19. Gonzalez, T., Ibarra, O.H., Sahni, S.: Bounds for LPT schedules on uniform processors. SIAM J. Comput. 6(1), 155–166 (1977)
20. Graham, R.L.: Bounds for certain multiprocessing anomalies. Bell Syst. Tech. J. 45(9), 1563–1581 (1966)
21. Hochbaum, D.S.: Various notions of approximations: Good, better, best and more. In: Hochbaum, D.S. (ed.) Approximation Algorithms. PWS Publishing Company, Boston (1997)
22. Hochbaum, D.S., Maass, W.: Approximation schemes for covering and packing problems in image processing and VLSI. J. ACM 32(1), 130–136 (1985)
23. Hochbaum, D.S., Shmoys, D.B.: Using dual approximation algorithms for scheduling problems: theoretical and practical results. J. ACM 34(1), 144–162 (1987)
24. Hochbaum, D.S., Shmoys, D.B.: A polynomial approximation scheme for scheduling on uniform processors: using the dual approximation approach. SIAM J. Comput. 17(3), 539–551 (1988)
25. Horowitz, E., Sahni, S.: Exact and approximate algorithms for scheduling nonidentical processors. J. ACM 23(2), 317–327 (1976)
26. Jansen, K.: An EPTAS for scheduling jobs on uniform processors: using an MILP relaxation with a constant number of integral variables. SIAM J. Discret. Math. 24(2), 457–485 (2010)
27. Jansen, K., Robenek, C.: Scheduling jobs on identical and uniform processors revisited. In: Proceedings of the 9th International Workshop on Approximation and Online Algorithms (WAOA), pp. 109–122 (2011)
28. Kannan, R.: Improved algorithms for integer programming and related lattice problems. In: Proceedings of the 15th Symposium Theory of Computing (STOC), pp. 193–206. ACM (1983)
29. Lenstra, H.W. Jr.: Integer programming with a fixed number of variables. Math. Oper. Res. 8(4), 538–548 (1983)
30. Leung, J.Y.-T., Wei, W.D.: Tighter bounds on a heuristic for a partition problem. Inf. Process. Lett. 56(1), 51–57 (1995)
31. Marx, D.: Parameterized complexity and approximation algorithms. Comput. J. 51(1), 60–78 (2008)
32. Woeginger, G.J.: A polynomial time approximation scheme for maximizing the minimum machine completion time. Oper. Res. Lett. 20(4), 149–154 (1997)