A PTAS for minimizing the total weighted completion time on identical parallel machines

MARTIN SKUTELLA* GERHARD J. WOEGINGER†

September 7, 1999

Abstract

We consider the problem of scheduling a set of n jobs on m identical parallel machines so as to minimize the weighted sum of job completion times. This problem is NP-hard in the strong sense. The best approximation result known so far was a $\frac{1}{2}(1+\sqrt{2})$–approximation algorithm that has been derived by Kawaguchi and Kyan back in 1986. The contribution of this paper is a polynomial time approximation scheme for this setting, which settles a problem that was open for a long time. Moreover, our result constitutes the first known approximation scheme for a strongly NP-hard scheduling problem with minsum objective.

Key words: scheduling theory, approximation algorithm, approximation scheme, worst-case ratio, combinatorial optimization

1 Introduction

The problem. We consider the following machine scheduling model. We are given a set J of n independent jobs that have to be scheduled on m identical parallel machines or processors. Each job $j \in J$ is specified by its positive processing requirement p_j and by its positive weight w_j. In a feasible schedule for J, every job $j \in J$ is processed for p_j time units on one of the m machines in an uninterrupted fashion. Every machine can process at most one job at a time, and every job can be processed on at most one machine at a time. The completion time of job j in some schedule is denoted by C_j. The goal is to minimize the total weighted completion time $\sum_{j\in J} w_j C_j$. In the standard classification scheme of Graham, Lawler, Lenstra, & Rinnooy Kan 1979, this scheduling problem is denoted by $\mathrm{P}\,|\,|\sum w_j C_j$ for m part of the input, and by $\mathrm{P}m\,|\,|\sum w_j C_j$ for constant m.

Complexity of the problem. For the special case of only one machine (i. e., $m = 1$) the problem can be solved in polynomial time by Smith's Ratio Rule: process the jobs in order of nonincreasing ratios $\frac{w_j}{p_j}$. Thus, for the single machine case, the 'importance' of a job is measured by its ratio. For a constant number $m \geq 2$ of machines, the problem is NP-hard in the ordinary sense and solvable in pseudopolynomial time. For m part of the input, the problem is NP-hard in the strong sense; see problem SS13 in Garey & Johnson 1979. The special case $\mathrm{P}\,|\,|\sum C_j$ with unit-weights (i. e., $w_j \equiv 1$) is solvable in polynomial time by sorting, see Conway, Maxwell, & Miller 1967.

Approximation algorithms. In this paper we are interested in how close one can approach an optimum solution to these NP-hard scheduling problems in polynomial time. Thus, our research focuses on *approximation algorithms* which efficiently construct schedules whose values are within a constant factor $\alpha > 1$ of the optimum solution value. The number α is called *performance guarantee* or *performance*

*Technische Universität Berlin, Fachbereich Mathematik, MA 6–1, Straße des 17. Juni 136, D–10623 Berlin, Germany, E-mail skutella@math.tu-berlin.de. Supported by DONET within the frame of the TMR Programme (contract number ERB FMRX-CT98-0202) while staying at C.O.R.E., Louvain-la-Neuve, Belgium, for the academic year 1998/99.

†Technische Universität Graz, Institut für Mathematik, Steyrergasse 30, A–8010 Graz, Austria, E-mail gwoegi@opt.math.tu-graz.ac.at. Supported by the START program Y43-MAT of the Austrian Ministry of Science.

ratio of the approximation algorithm. A family of polynomial time approximation algorithms with performance guarantee $1+\varepsilon$ for all fixed $\varepsilon > 0$ is called a *polynomial time approximation scheme* (PTAS). If the running times of the approximation algorithms are even bounded by a polynomial in the input size and $\frac{1}{\varepsilon}$, then these algorithms build a *fully polynomial time approximation scheme* (FPTAS). It is known that unless P=NP, a strongly NP-hard optimization problem cannot possess an FPTAS, see Garey & Johnson 1979.

Known approximation results. Sahni (1976) gives a FPTAS for the weakly NP-hard scheduling problem $\mathrm{P}m\,|\,|\sum w_j C_j$ with fixed m. Kawaguchi & Kyan (1986) analyze list scheduling in order of nonincreasing ratios $\frac{w_j}{p_j}$ on identical parallel machines. They prove a performance ratio $\frac{1}{2}(1+\sqrt{2})$ for the strongly NP-hard problem $\mathrm{P}\,|\,|\sum w_j C_j$. Till now, this was the best approximation result for this problem. Alon, Azar, Woeginger, & Yadid (1998) study scheduling problems on identical parallel machines with various objective functions that solely depend on the *machine* completion times. In particular, they give a polynomial time approximation scheme for the problem of minimizing $\sum_{i=1}^{m} {M_i}^2$ where M_i denotes the finishing time of machine i. By rewriting the objective function in an appropriate way, this result implies a PTAS for the problem of minimizing the weighted sum of job completion times if all job ratios $\frac{w_j}{p_j}$, $j \in J$, are equal.

Generally speaking, the approximability of scheduling problems with total job completion time objective (so-called *minsum* scheduling problems) is not well-understood. Some minsum problems can be solved in polynomial time using straightforward algorithms (like the single machine version of the problem $\mathrm{P}m\,|\,|\sum w_j C_j$). Some weakly NP-hard minsum problems allow an FPTAS based on dynamic programming formulations (see, e. g., Sahni 1976 and Woeginger 1998). Some minsum problems do not have a PTAS unless P=NP (Hoogeveen, Schuurman, & Woeginger 1998). Some of these problems cannot even be approximated in polynomial time within a constant factor (like minimizing total flow time, see Kellerer, Tautenhahn, & Woeginger 1996 and Leonardi & Raz 1997). Some minsum problems have constant factor approximation algorithms that are based on rounding and/or transforming and/or manipulating the solutions of preemptive relaxations or relaxations of integer programming formulations (see, e. g., Phillips, Stein, & Wein 1995, Hall, Schulz, Shmoys, & Wein 1997, or Skutella 1998); due to the integrality gap, these approaches can never yield a PTAS. However, there is not a single PTAS known for a strongly NP-hard minsum scheduling problem.

Contribution of this paper. Our contribution is a polynomial time approximation scheme for the general problem $\mathrm{P}\,|\,|\sum w_j C_j$ of minimizing the total weighted completion time on identical parallel machines. This result is derived in two steps. In the first step, we give a PTAS for the special case where the largest job ratio is only a constant factor away from the smallest job ratio. This result is derived by modifying and by generalizing a technique of Alon et al. 1998. In the second step, we construct a PTAS for $\mathrm{P}\,|\,|\sum w_j C_j$ in its full generality. The main idea is to partition the jobs into subsets according to their ratios such that near optimal schedules can be computed for all subsets; the key observation is that these schedules can be concatenated without too much loss in the overall performance guarantee.

Our result yields the first polynomial time approximation scheme for a strongly NP-hard scheduling problem with minsum objective. It confirms a conjecture of Hoogeveen, Schuurman, & Woeginger 1998, it solves an open problem posed in Alon et al. 1998, and it improves on the result by Kawaguchi & Kyan 1986.

Recently, several research groups have found polynomial-time approximation schemes for problems with release dates such as $1|\,r_j\,|\sum C_j$ and $\mathrm{P}\,|\,r_j\,|\sum w_j C_j$. We refer to the resulting joint conference proceedings publication (Afrati et al. 1999) for details.

Organization of the paper. Section 2 contains preliminaries which will be used throughout the paper. In Section 3 we give an approximation scheme for the special case that the ratios of jobs are within a constant range. Finally, in Section 4 we present the approximation scheme for arbitrary instances of $\mathrm{P}\,|\,|\sum w_j C_j$.

2 Preliminaries

By Smith's Ratio Rule, it is locally optimal to schedule the jobs that have been assigned to a machine in order of nonincreasing ratios $\frac{w_j}{p_j}$; the proof is a simple exchange argument. Throughout the paper, we restrict to schedules meeting this property; thus, whenever we consider a schedule, we always implicitly assume that it is locally optimal in this sense. For a given schedule let $\Gamma_j := C_j - \frac{1}{2}p_j$, for $j \in J$, such that

$$\sum_{j\in J} w_j C_j \;=\; \sum_{j\in J} w_j \Gamma_j + \frac{1}{2}\sum_{j\in J} w_j p_j \;. \tag{1}$$

Notice that the second term on the right hand side is nonnegative and does not depend on the specific schedule. Therefore, for each $\varepsilon > 0$, a $(1+\varepsilon)$–approximation algorithm for the problem to minimize the function $\sum_{j\in J} w_j\Gamma_j$ is also a $(1+\varepsilon)$–approximation algorithm for minimizing the total weighted completion time. In fact, in Section 3 it will be more convenient to consider the objective function $\sum_{j\in J} w_j\Gamma_j$ and we will give a polynomial time approximation scheme for the problem to minimize this function there.

We use the following notation: For a subset of jobs $J' \subseteq J$, let $p(J') := \sum_{j\in J'} p_j$ denote the total processing time of the jobs in J'. Moreover, denote the average machine load caused by the jobs in J' by $L(J') := \frac{1}{m}p(J')$; for $J' = J$ we use the simpler notation $L := L(J)$. We start with the following observation: Consider a subset of jobs $J' \subseteq J$ with $\frac{w_j}{p_j} = \rho$ for all $j \in J'$; if the jobs in J' are consecutively scheduled on a machine in an arbitrary order starting at time τ, their contribution to the objective function is given by

$$\sum_{j\in J'} w_j\Gamma_j \;=\; \frac{\rho}{2}\cdot p^2(J') + \rho\cdot\tau\cdot p(J') \;. \tag{2}$$

This observation together with Smith's Ratio Rule leads to the following lemma which generalizes a result of Eastman, Even, & Isaacs 1964.

Lemma 2.1. *Let $\rho_1 > \rho_2 > \cdots > \rho_q$ denote the different job ratios $\frac{w_j}{p_j}$, $j \in J$. Moreover, for $1 \le h \le q$, let $J_h := \{j \in J \mid \frac{w_j}{p_j} = \rho_h\}$. For a given schedule, denote the subset of J_h that is being assigned to machine i by $J_{h,i}$. Then, the value of the schedule is given by*

$$\sum_{j\in J} w_j\Gamma_j \;=\; \frac{1}{2}\sum_{h=1}^{q}\rho_h\sum_{i=1}^{m}p^2(J_{h,i}) + \sum_{1\le h<\ell\le q}\rho_\ell\sum_{i=1}^{m}p(J_{h,i})p(J_{\ell,i}) \;. \tag{3}$$

Proof. On each machine i and for each $1 \le \ell \le q$, the jobs in $J_{\ell,i}$ are scheduled consecutively starting at time $\sum_{h=1}^{\ell-1} p(J_{h,i})$. The result thus follows from (2). □

Notice that the right hand side of (3) only depends on the total processing times of the sets $J_{h,i}$, but not on the structure of these sets. In the analysis of the approximation scheme we will make use of the following lower bound on the value of an optimal schedule:

Lemma 2.2. *Using the same notation as in Lemma 2.1, the value of an arbitrary schedule is bounded from below by*

$$\sum_{j\in J} w_j C_j \;\ge\; \frac{m}{2}\sum_{h=1}^{q}\rho_h L^2(J_h) \;.$$

Proof. By (1) and Lemma 2.1 we get

$$\sum_{j\in J} w_j C_j \;\ge\; \frac{1}{2}\sum_{h=1}^{q}\rho_h\sum_{i=1}^{m}p^2(J_{h,i}) \;\ge\; \frac{m}{2}\sum_{h=1}^{q}\rho_h L^2(J_h) \;.$$

The second inequality follows from the convexity of the function $x \mapsto x^2$. □

Throughout the paper, $\mathbb{N}$ denotes the set of positive integers; the set of nonnegative integers is denoted by $\mathbb{N}_0$.

3 The approximation scheme for a constant range of ratios

In this section we consider the problem to minimize $\sum_{j\in J} w_j\Gamma_j$. Therefore, whenever we refer to the 'value' of a schedule we mean $\sum_{j\in J} w_j\Gamma_j$. We give a polynomial time approximation scheme for instances with bounded weight to length ratios of jobs, i. e., where all job ratios $\frac{w_j}{p_j} \in [\rho R, R]$ for an arbitrary $R > 0$ and a real constant $0 < \rho \le 1$ that does not depend on the input. First notice that by rescaling the weights of jobs we can restrict to the case $R = 1$.

3.1 Structural insights

Let $0 < \delta < 1$ be an arbitrary real constant and choose a corresponding constant $\Delta \in \mathbb{N}$ with $\delta^{\Delta+1} < \rho$. If we round up the weights of jobs such that the ratio of each job attains the nearest integer power of δ, the value of an optimal schedule increases at most by a factor $\frac{1}{\delta}$. Since the constant $\delta < 1$ can be chosen arbitrarily close to 1, we may restrict to instances of $P \mid\mid \sum w_jC_j$ where all job ratios $\frac{w_j}{p_j}$ are of the form δ^h with $h \in \{0, 1, \ldots, \Delta\}$. We use the following notation in this section: For $h = 0, 1, \ldots, \Delta$, denote by J_h the class of all jobs $j \in J$ with $\frac{w_j}{p_j} = \delta^h$. The completion time of machine i in some schedule is denoted by M_i.

Lemma 3.1. *Let $f := 1 + \frac{1}{\delta^\Delta}$; if in an optimal schedule $M_i > fM_{i'}$ for a pair of machines i, i', then machine i processes exactly one job.*

Proof. Let j denote the last job on machine i. Observe that j must start at or before time $M_{i'}$; otherwise, moving j to the end of machine i' would decrease Γ_j and hence the value of the schedule. Thus, we get $p_j \ge M_i - M_i' > \frac{1}{\delta^\Delta}M_{i'}$. By contradiction, assume that j is not the first job on machine i; denote the first job by k. Let W_i and $W_{i'}$ denote the sum of the weights of jobs scheduled on machine i and i', respectively. Since $w_j \ge \delta^\Delta p_j$ and $p_j > \frac{1}{\delta^\Delta}M_{i'}$, we get $W_i - w_k > M_{i'}$; moreover, $W_{i'} \le M_{i'}$ since the weight to length ratios of all jobs are at most 1. Thus, removing job k from machine i and inserting it at the beginning of machine i' changes the value of the schedule by $p_k(W_{i'} - W_i + w_k) < 0$, contradicting optimality. □

For the following corollary notice that there is at least one machine i with $M_i \le L$ in every schedule.

Corollary 3.2. *If a job j has size $p_j \ge fL$, then j occupies a machine of its own in any optimal schedule. If no job has size greater than fL, then the completion time of each machine is at most fL in any optimal schedule.*

As a result of Corollary 3.2, we can iteratively reduce a given instance: as long as a job j of size $p_j \ge fL$ is available, remove it from the set of jobs J, assign it to a machine of its own, and decrease the number of machines by one. In the following we can thus restrict to instances of the following form:

Assumption 3.3. *The processing time p_j of each job j as well as the completion time M_i of any machine i in an optimal schedule are bounded from above by fL.*

3.2 Rounding the instance

We define a simplified, rounded version of the input for which we can compute an optimal schedule in polynomial time. Moreover, under some assumption that we will specify later, an optimal solution to the rounded instance will lead to a near optimal solution to the original instance. The rounding is based on the positive integral constant λ which will later be chosen to be 'sufficiently large'.

The rounding is done for every class J_h, $0 \le h \le \Delta$, separately. We will replace the jobs in J_h by new jobs, with slightly different processing times; however, the length to weight ratio will stay at δ^h.

- Every 'big' job j in the original instance with $p_j > \frac{L}{\lambda}$, is replaced by a corresponding rounded job $j^\#$ whose processing time $p_j^\#$ equals p_j rounded up to the next integer multiple of $\frac{L}{\lambda^2}$. The weight $w_j^\#$ of the rounded job equals $\delta^h p_j^\#$. Note that for the rounded processing time $p_j \le p_j^\# \le \frac{\lambda+1}{\lambda}p_j$ holds.

- Denote by S_h the total processing time of the 'small' jobs in J_h whose processing times are not greater than $\frac{L}{\lambda}$. Denote by $S_h^\#$ the value of S_h rounded up to the next integer multiple of $\frac{L}{\lambda}$. Then the rounded instance contains $S_h^\# \frac{\lambda}{L} + m$ new jobs, each of length $\frac{L}{\lambda}$. The weight of every new job equals $\delta^h \frac{L}{\lambda}$.

Note that the total number of jobs in the rounded instance can be bounded by $n + (\Delta + 1)m$. We get the following lemma.

Lemma 3.4. *By replacing the rounded jobs with their unrounded counterparts, an arbitrary schedule for the transformed instance induces a schedule of smaller or equal value for the original instance.*

Proof. First notice that the rounded big jobs $j^\#$ can be replaced by their unrounded counterparts without increasing the value of the schedule.

Consider a set of small jobs S_h. We can assume without loss of generality that, on each machine, the corresponding rounded jobs of size $\frac{L}{\lambda}$ are processed consecutively in the given schedule. We remove those jobs from the machines and pack the jobs in S_h greedily into the m resulting intervals of idle time. This is possible since the number of rounded jobs was chosen large enough. Again, the value of the schedule is not increased by this replacement. □

The rounded instance can be solved in polynomial time, e. g., by dynamic programming. However, we use a generalization of an alternative approach of Alon, Azar, Woeginger, & Yadid 1998; we formulate the problem as an integer linear program whose dimension is bounded by a constant. By Assumption 3.3 and by construction, the size $p_j^\#$ of each job j is bounded by

$$p_j^\# \;\leq\; fL + \frac{L}{\lambda^2} \;.$$

Therefore, since all job sizes are integer multiples of $\frac{1}{\lambda^2}L$, there is at most a constant number Π of possible sizes $k\frac{L}{\lambda^2}$, $k = 1, \ldots, \Pi$. Moreover, since the number of possible length to weight ratios is bounded by the constant $\Delta + 1$, the total number of different types of jobs is bounded by the constant $\Pi(\Delta + 1)$. For $k = 1, \ldots, \Pi$ and $h = 0, \ldots, \Delta$, let $n_{k,h}$ denote the number of jobs of size $k\frac{L}{\lambda^2}$ and ratio δ^h; we define the vector $\mathbf{n} := (n_{1,0}, n_{1,1}, \ldots, n_{\Pi,\Delta})$. An assignment of jobs to one machine is given by a vector $\mathbf{u} = (u_{1,0}, u_{1,1}, \ldots, u_{\Pi,\Delta})$ where $u_{k,h}$ is the number of jobs of size $k\frac{L}{\lambda^2}$ and ratio δ^h that are assigned to the machine. The completion time of the machine is given by $M(\mathbf{u}) = \sum_{k=1}^{\Pi} \sum_{h=0}^{\Delta} u_{k,h} \cdot k\frac{L}{\lambda^2}$. Moreover, let $c(\mathbf{u})$ denote the contribution to the objective function of a machine that is scheduled according to $\mathbf{u}$.

Let $L^\# := \frac{1}{m}\sum_j p_j^\#$ denote the average machine load for the transformed instance; observe that by construction

$$L^\# \;\leq\; \frac{\lambda+1}{\lambda}L + \frac{(\Delta+1)(m+1)}{\lambda m}L \;\leq\; \frac{\lambda+2\Delta+3}{\lambda}L \;.$$

Thus, by Assumption 3.3 we can bound the completion time of each machine by $f\frac{\lambda+2\Delta+3}{\lambda}L$. Denote by U the set of vectors $\mathbf{u}$ with $M(\mathbf{u}) \leq f\frac{\lambda+2\Delta+3}{\lambda}L$. For a vector $\mathbf{u} \in U$, each entry $u_{k,h}$ is bounded by a number that only depends on the constants λ, Δ, f and is thus independent of the input. Therefore the set U is of constant size.

We can now formulate the problem of finding an optimal schedule as an integer linear program with a constant number of variables. For each vector $\mathbf{u} \in U$ we introduce a variable $x_\mathbf{u}$ which denotes the number of machines that are assigned jobs according to $\mathbf{u}$. An optimal schedule is then given by the following program:

$$\begin{aligned}
\min\ & \sum_{\mathbf{u}\in U} x_\mathbf{u} \cdot c(\mathbf{u}) \\
\text{subject to}\ & \sum_{\mathbf{u}\in U} x_\mathbf{u} = m \\
& \sum_{\mathbf{u}\in U} x_\mathbf{u} \cdot \mathbf{u} = \mathbf{n} \\
& x_\mathbf{u} \in \mathbb{N}_0 \quad \text{for all } \mathbf{u} \in U
\end{aligned}$$

It has been shown by Lenstra (1983) that an integer linear program in constant dimension can be solved in polynomial time.

3.3 Proving near optimality

By Lemma 3.4 it remains to show that the value of an optimal schedule for the rounded instance is at most a factor of $(1+\varepsilon)$ above the optimal objective value of the original instance. We will prove this under the following assumption on the original instance, and afterwards we will demonstrate how to get rid of the assumption:

Assumption 3.5. *There exists an optimal schedule of the following form: The completion time M_i of every machine i fulfills the inequality $\frac{L}{f} \le M_i \le fL$.*

In order to achieve the desired precision, we now choose the integer λ sufficiently large to fulfill the inequality

$$\Big(\frac{\lambda+1}{\lambda}\Big)^2 + \frac{4f^2}{\lambda^2\delta^\Delta}(f\lambda + f + \Delta + 1)(\Delta+1) \;\le\; 1+\varepsilon \;. \tag{4}$$

Since Δ, δ, f, and ε are constants that do not depend on the input, also λ is constant and independent of the input.

Lemma 3.6. *Under Assumption 3.5, the optimal objective value of the rounded instance is at most a factor of $(1+\varepsilon)$ above the optimal objective value of the original instance.*

Proof. Take an optimal schedule for the original instance as described in Assumption 3.5. Consider some fixed machine i with finishing time M_i in this optimal schedule. For $h = 0, 1, \ldots, \Delta$, let J'_h denote the subset of J_h processed on machine i. Note that

$$\frac{L}{f} \;\le\; \sum_{0\le h\le\Delta} p(J'_h) \;\le\; fL \;. \tag{5}$$

By Lemma 2.1, the contribution of machine i to the objective function is given by

$$\sum_{j\in J'} w_j\Gamma_j \;=\; \frac{1}{2}\sum_{0\le h\le\Delta} \delta^h \cdot p^2(J'_h) \;+\; \sum_{0\le h<\ell\le\Delta} \delta^\ell \cdot p(J'_h)p(J'_\ell) \;. \tag{6}$$

Replace every big job j by its corresponding rounded big job $j^\#$. This may increase every $p(J'_h)$ by a multiplicative factor of at most $\frac{\lambda+1}{\lambda}$. For every J'_h, denote by s_h the total size of its small jobs. Round s_h up to the next integer multiple of $\frac{L}{\lambda}$, increase it by one, and replace the small jobs in J'_h by an appropriate number of jobs of size $\frac{L}{\lambda}$ and length to weight ratio δ^h. This may increase $p(J'_h)$ by an additive factor of at most $\frac{2L}{\lambda}$. Note that by repeating this replacement procedure for all machines, one can accommodate all jobs in the rounded instance.

Denote by K_h the resulting set of rounded jobs that are assigned to machine i and have length to weight ratio δ^h. We have for $h = 0, 1, \ldots, \Delta$ that

$$p(K_h) \;\le\; \frac{\lambda+1}{\lambda}p(J'_h) + \frac{2L}{\lambda} \;.$$

Now compare the first term on the right hand side of (6) for J'_h to the corresponding term for the jobs in K_h:

$$\begin{aligned}
\frac{1}{2}\sum_{0\le h\le\Delta} \delta^h \cdot p^2(K_h) \;&\le\; \frac{1}{2}\sum_{0\le h\le\Delta} \delta^h \cdot \Big(\frac{\lambda+1}{\lambda}p(J'_h) + \frac{2L}{\lambda}\Big)^2 \\
&\le\; \frac{1}{2}\Big(\frac{\lambda+1}{\lambda}\Big)^2 \sum_{0\le h\le\Delta} \delta^h \cdot p^2(J'_h) \;+\; \sum_{0\le h\le\Delta} \delta^h \Big(2\frac{\lambda+1}{\lambda^2}p(J'_h)L + \frac{2L^2}{\lambda^2}\Big) \\
&\le\; \frac{1}{2}\Big(\frac{\lambda+1}{\lambda}\Big)^2 \sum_{0\le h\le\Delta} \delta^h \cdot p^2(J'_h) \;+\; 2\frac{\lambda+1}{\lambda^2}fL^2 \;+\; \frac{2}{\lambda^2}(\Delta+1)L^2 \;. \qquad (7)
\end{aligned}$$

In the last inequality, we applied $\delta \leq 1$ together with the second inequality in (5). In a similar way, we compare the second term on the right hand side of (6) for J'_h to the corresponding term for the jobs in K_h:

$$
\begin{aligned}
&\sum_{0\leq h<\ell\leq\Delta} \delta^\ell \cdot p(K_h)p(K_\ell) \\
&\leq \sum_{0\leq h<\ell\leq\Delta} \delta^\ell \cdot \Big(\frac{\lambda+1}{\lambda}p(J'_h) + \frac{2L}{\lambda}\Big)\cdot\Big(\frac{\lambda+1}{\lambda}p(J'_\ell) + \frac{2L}{\lambda}\Big) \\
&\leq \Big(\frac{\lambda+1}{\lambda}\Big)^2 \sum_{0\leq h<\ell\leq\Delta} \delta^\ell \cdot p(J'_h)p(J'_\ell) \;+\; \frac{\lambda+1}{\lambda}\cdot\frac{2L}{\lambda}\cdot\Delta \sum_{0\leq h\leq\Delta} p(J'_h) \;+\; \frac{\Delta(\Delta+1)}{2}\cdot\frac{4L^2}{\lambda^2} \\
&\leq \Big(\frac{\lambda+1}{\lambda}\Big)^2 \sum_{0\leq h<\ell\leq\Delta} \delta^\ell \cdot p(J'_h)p(J'_\ell) \;+\; 2\Delta\frac{\lambda+1}{\lambda^2}fL^2 \;+\; \Delta(\Delta+1)\frac{2L^2}{\lambda^2}\ . \qquad (8)
\end{aligned}
$$

Since $\delta \leq 1$ and because of the first inequality in (5), we get that the right hand side of (6) fulfills:

$$
\frac{1}{2}\sum_{0\leq h\leq\Delta} \delta^h \cdot p^2(J'_h) + \sum_{0\leq h<\ell\leq\Delta} \delta^\ell \cdot p(J'_h)p(J'_\ell) \;\geq\; \frac{\delta^\Delta}{2}\left(\sum_{0\leq h\leq\Delta} p(J'_h)\right)^2 \;\geq\; \frac{\delta^\Delta}{2}\cdot\frac{L^2}{f^2}\ .
$$

Putting this together with (7), (8), and (4), a short calculation shows that the objective value for the rounded instance is at most a factor of $(1+\varepsilon)$ above the optimal objective value for the original instance. □

Finally, we consider the case when Assumption 3.5 may be violated. By Assumption 3.3 we can restrict to the case that only the lower bound $\frac{1}{f}L$ can be violated for some machine completion time. As a result of Lemma 3.1 we get:

Corollary 3.7. *If in an optimal schedule $M_i < \frac{L}{f}$ for some machine i, then every machine i' with $M_{i'} \geq L$ processes only one job; in particular, there exists a job j of size $p_j \geq L$ and every such job occupies a machine of its own.*

In the following we assume that there exists a job j of size $p_j \geq L$; otherwise, Assumption 3.5 is true by Corollary 3.7 and we are finished. Unfortunately, we do not know in advance whether or not $M_i < \frac{L}{f}$ for some machine i in an optimal schedule. Therefore we take both possibilities into account and compute two schedules for the given instance such that the better one is guaranteed to be a $(1+\varepsilon)$–approximation by Corollary 3.7.

On the one hand, we compute an optimal schedule for the rounded instance and turn it into a feasible schedule for the original instance as described in Lemma 3.4. On the other hand, we assign each job j with $p_j \geq L$ to a machine, remove those jobs from the instance, and decrease the number of machines by the number of removed jobs. For the reduced instance, we can recursively compute a $(1+\varepsilon)$–approximation by again taking the better of two schedules. Notice that in each recursion step the number of machines is decreased by at least one; thus, after at most $m-1$ steps we arrive at a trivial problem that can be solved to optimality in polynomial time.

We can now state the main result of this section:

Theorem 3.8. *There exists a PTAS for the special case of the problem $\mathrm{P}\,|\,|\sum w_jC_j$ where all job ratios are in a constant range $[\rho R, R]$ for an arbitrary $R > 0$ and a real constant $0 < \rho \leq 1$ that does not depend on the input.*

4 The polynomial time approximation scheme

In this section we return to the objective function $\sum_{j\in J} w_jC_j$ and present the approximation scheme for arbitrary instances of $\mathrm{P}\,|\,|\sum w_jC_j$. The main idea for deriving this result is to partition the set of jobs into subsets according to their ratios $\frac{w_j}{p_j}$. The ratios of all jobs in one subset are within a constant range such that for each subset a near optimal schedule can be computed within arbitrary precision in polynomial time, see Theorem 3.8. In a second step, these 'partial schedules' are concatenated in order of nonincreasing job ratios such that Smith's Ratio Rule is obeyed on each machine.

For the sake of a more accessible analysis, we first present a randomized variant of the approximation scheme and discuss its derandomization later. Throughout this section we assume that $\frac{w_j}{p_j} \leq 1$ for all jobs $j \in J$; otherwise, all weights of jobs can be rescaled by the inverse of the maximal ratio.

4.1 The randomized approximation scheme

The partitioning step. Let Δ be a positive integer and let $\delta := \frac{1}{\Delta}$; later we will choose Δ large such that δ gets small. The partitioning of the set of jobs J is performed in two steps. The first step computes a fine partition which is then randomly turned into a rougher partition in the second step.

1. For $h \in \mathbb{N}$ let $J(h) := \left\{ j \in J \mid \delta^h < \frac{w_j}{p_j} \leq \delta^{h-1} \right\}$.
2. Draw q uniformly at random from $\{1, 2, \ldots, \Delta\}$; set $I_0^q := \{1, 2, \ldots, q-1\}$ and, for $s \in \mathbb{N}$,

$$I_s^q := \{(s-1) \cdot \Delta + q, \ldots, s \cdot \Delta - 1 + q\} \ ;$$

   for $s \in \mathbb{N}_0$, let $J_s^q := \bigcup_{h \in I_s^q} J(h)$.

Notice that for fixed q the number of nonempty subsets J_s^q, $s \in \mathbb{N}_0$, is bounded by n. Of course, we only take those subsets into consideration in the algorithm. The intuition for step 2 is to compensate the undesired property of the fine partition computed in step 1 that jobs with similar ratio may lie in different subsets of the computed partition. The random choice in step 2 assures that the probability for those jobs to lie in different subsets of the rough partition is small, i. e., in $O\left(\frac{1}{\Delta}\right)$.

Computing partial schedules. The quotient of the biggest and the smallest ratio of jobs in a subset J_s^q is bounded by the constant δ^{Δ}. Thus, by Theorem 3.8, we can compute in polynomial time for all nonempty sets of jobs J_s^q near optimal m–machine schedules of value $Z_s^q \leq (1+\delta) \cdot Z_s^{q*}$, where Z_s^{q*} denotes the value of an optimal schedule for J_s^q.

The concatenation step. In the final step of the algorithm these partial schedules are concatenated. One possibility is to do it machine-wise: On machine i, all jobs that have been assigned to i in the partial schedules are processed according to Smith's Ratio Rule. However, this deterministic concatenation can lead to an undesired unbalance of load on the machines. It might for example happen that each subset of jobs J_s^q consists of at most one job which is always assigned to machine 1 in the corresponding partial schedule. Thus, concatenating the partial schedules as proposed above would leave all but one machine idle.

Therefore, we first randomly and uniformly permute the numbering of machines in each partial schedule and then apply the machine-wise concatenation described above. In this randomly generated schedule the probability for two jobs from different subsets to be processed on the same machine is equal to $\frac{1}{m}$ such that one can expect an appropriately balanced machine assignment.

4.2 The analysis of the randomized approximation scheme

The analysis is based on the observation that the value of the computed schedule is composed of the sum of the values of the partial schedules plus the additional cost caused by the delay of jobs in the concatenation step. It is easy to see that the sum of the values of the near optimal partial schedules cannot substantially exceed the value of an optimal schedule.

The key insight for the analysis is that the delay of jobs in one subset caused by another subset in the concatenation step can essentially be neglected. One reason is that the delayed jobs usually (i. e., with high probability) have much smaller ratio and are thus less important than the jobs which cause the delay. On the other hand, if there are too many 'unimportant' jobs to be neglected, then the total weighted completion time of the corresponding near optimal partial schedule must be large compared to the delay caused by the important jobs.

The following lemma provides two lower bounds on the value of an optimal schedule Z^*. To simplify notation, let $L(h) := L(J(h))$ for $h \in \mathbb{N}$.

Lemma 4.1. *For each $q \in \{1, \ldots, \Delta\}$, the value Z^* of an optimal schedule is bounded by*

$$Z^* \;\geq\; \sum_{s \in \mathbb{N}_0} Z_s^{q*} \qquad \textit{and} \qquad Z^* \;\geq\; \frac{m}{2} \sum_{h \in \mathbb{N}} \delta^h \cdot L^2(h) \;\;. \tag{9}$$

Proof. Take an optimal schedule for the set of jobs J and denote the completion time of job j in this schedule by C_j^*. This yields

$$Z^* \;=\; \sum_{j \in J} w_j C_j^* \;=\; \sum_{s \in \mathbb{N}_0} \sum_{j \in J_s^q} w_j C_j^* \;\geq\; \sum_{s \in \mathbb{N}_0} Z_s^{q*}$$

since the completion times C_j^* also define a feasible schedule for each subset of jobs J_s^q. In order to prove the second lower bound, first observe that the value of an optimal schedule decreases if we round the weights of jobs $j \in J(h)$ to $w_j = \delta^h p_j$, for $h \in \mathbb{N}$. The result then follows from Lemma 2.2. ☐

The next step in the analysis is to determine an expression for the expected value of the computed schedule.

Lemma 4.2. *The expected value of the computed schedule is given by*

$$\mathbb{E}\Big[\sum_{j \in J} w_j C_j\Big] \;=\; \mathbb{E}\Big[\sum_{s \in \mathbb{N}_0} Z_s^q\Big] + \sum_{h<k} \min\Big\{1, \frac{k-h}{\Delta}\Big\} L(h) \sum_{j \in J(k)} w_j \;\;. \tag{10}$$

Proof. We first keep q fixed and analyze the conditional expectation $\mathbb{E}_q\big[\sum_{j \in J} w_j C_j\big]$. This conditional expectation is equal to the sum of the values of the partial schedules $\sum_{s \in \mathbb{N}_0} Z_s^q$ plus the expected cost caused by the delay of jobs in the concatenation step.

The expected delay of an arbitrary job $j \in J$ can be determined as follows. Let $t \in \mathbb{N}_0$ and $k \in I_t^q$ such that $j \in J(k) \subseteq J_t^q$. Then, the expected delay of job j is equal to the expected load caused by jobs from $\bigcup_{s=0}^{t-1} J_s^q$ on the machine which job j is being processed on. Since the machines are permuted uniformly at random in the concatenation step, the expected load is equal to the average load

$$\sum_{s=0}^{t-1} \sum_{h \in I_s^q} L(h) \;=\; \sum_{h=1}^{k-1} \eta_{h,k}^q \cdot L(h)$$

where $\eta_{h,k}^q = 1$ if h and k lie in different sets of indices I_r^q, $r \in \mathbb{N}_0$, and $\eta_{h,k}^q = 0$, otherwise. As a consequence, for fixed q, the conditional expectation of the total weighted completion time can be written as:

$$\mathbb{E}_q\Big[\sum_{j \in J} w_j C_j\Big] \;=\; \sum_{s \in \mathbb{N}_0} Z_s^q + \sum_{h<k} \eta_{h,k}^q L(h) \sum_{j \in J(k)} w_j \tag{11}$$

Notice that for randomly chosen q the expected value of $\eta_{h,k}^q$ is equal to the probability that h and k lie in different sets of indices I_r^q, $r \in \mathbb{N}_0$. By construction of the sets of indices I_r^q, this probability is equal to $\frac{k-h}{\Delta}$ for $k - h \leq \Delta$ and it is 1 for $k - h > \Delta$. The result thus follows from (11). ☐

The following theorem contains the main result of this subsection.

Theorem 4.3. *For a given $0 < \varepsilon \leq 1$ let $\Delta := \lceil \frac{36}{\varepsilon^2} \rceil$. Then, the expected value of the computed schedule is bounded by $1 + \varepsilon$ times the value of an optimal solution, i. e.,*

$$\mathbb{E}\Big[\sum_{j \in J} w_j C_j\Big] \;\leq\; (1 + \varepsilon) \cdot Z^* \;\;.$$

Proof. We compare the expected value given in Lemma 4.2 to the lower bounds on the value of an optimal schedule given in Lemma 4.1. Since, by construction of the algorithm, $Z_s^q \leq (1 + \delta) \cdot Z_s^{q*}$, for each q, and

$\sum_{s\in\mathbb{N}_0} Z_s^{q*} \le Z^*$ by Lemma 4.1, the first term on the right hand side of (10) can be bounded from above by

$$\mathbb{E}\Big[\sum_{s\in\mathbb{N}_0} Z_s^q\Big] \;\le\; (1+\delta)\cdot Z^* \;\;.$$

In order to bound the second term on the right hand side of (10), first observe that for each $k \in \mathbb{N}$

$$\sum_{j\in J(k)} w_j \;\le\; \sum_{j\in J(k)} p_j \cdot \delta^{k-1} \;=\; m\cdot L(k)\cdot \delta^{k-1} \;\;.$$

Thus, the inequality for the geometric and the arithmetic mean yields

$$L(h)\sum_{j\in J(k)} w_j \;\le\; m\cdot L(h)\cdot L(k)\cdot \delta^{k-1} \;\le\; \delta^{\frac{k-h}{2}-1}\cdot \frac{m}{2}\Big(L^2(h)\cdot\delta^h + L^2(k)\cdot \delta^k\Big) \;\;. \tag{12}$$

Notice that (12) bounds the cost for the possible delay of jobs in $J(k)$ caused by jobs in $J(h)$ in terms of the lower bounds on optimal schedules for $J(h)$ and $J(k)$ in (9). In particular, if $k-h\ge 3$, then the cost for the delay is small compared to the sum of the values of optimal schedules for $J(h)$ and $J(k)$. For the cases $k=h+1$ and $k=h+2$, however, this cost may be too large to be neglected. This is the point where we will make use of the random choice of q.

To be more precise, we divide the sum over all pairs $h<k$ in the second term of (10) into three partial sums $\Sigma_1+\Sigma_2+\Sigma_3$. The first partial sum Σ_1 takes all pairs with $k=h+1$ into account and can thus be bounded by

$$\begin{aligned}
\Sigma_1 \;&\le\; \frac{1}{\sqrt{\delta}}\cdot\frac{m}{2\Delta}\sum_{h\in\mathbb{N}}\Big(L^2(h)\cdot\delta^h + L^2(h+1)\cdot\delta^{h+1}\Big) && \text{by (12)}\\
&\le\; \sqrt{\delta}\cdot m\sum_{h\in\mathbb{N}} L^2(h)\cdot\delta^h\\
&\le\; 2\sqrt{\delta}\cdot Z^* && \text{by (9).}
\end{aligned}$$

The second partial sum Σ_2 takes all pairs with $k=h+2$ into account and can be bounded using the same arguments

$$\begin{aligned}
\Sigma_2 \;&\le\; \frac{m}{\Delta}\sum_{h\in\mathbb{N}}\Big(L^2(h)\cdot\delta^h + L^2(h+2)\cdot\delta^{h+2}\Big) && \text{by (12)}\\
&\le\; 2\delta\cdot m\sum_{h\in\mathbb{N}} L^2(h)\cdot\delta^h\\
&\le\; 4\delta\cdot Z^* && \text{by (9).}
\end{aligned}$$

Finally, the third partial sum Σ_3 takes all pairs with $k\ge h+3$ into account. In this case we bound the term $\min\{1,\frac{k-h}{\Delta}\}$ by 1 (i. e., we do not make use of the random choice of q) and get

$$\begin{aligned}
\Sigma_3 \;&\le\; \frac{m}{2}\sum_{h\le k-3} L^2(h)\cdot\delta^h\cdot\delta^{\frac{k-h}{2}-1} + \frac{m}{2}\sum_{h\le k-3} L^2(k)\cdot\delta^k\cdot\delta^{\frac{k-h}{2}-1} && \text{by (12)}\\
&=\; \frac{m}{2}\sum_{h\in\mathbb{N}} L^2(h)\cdot\delta^h\sum_{k=h+3}^{\infty}\delta^{\frac{k-h}{2}-1} + \frac{m}{2}\sum_{k\in\mathbb{N}} L^2(k)\cdot\delta^k\sum_{h=1}^{k-3}\delta^{\frac{k-h}{2}-1}\\
&\le\; m\sum_{h\in\mathbb{N}} L^2(h)\cdot\delta^h\sum_{\ell=1}^{\infty}\sqrt{\delta}^{\,\ell}\\
&\le\; 2\cdot\frac{\sqrt{\delta}}{1-\sqrt{\delta}}\cdot Z^* && \text{by (9).}
\end{aligned}$$

Putting the results together, the expected value of the computed schedule is bounded by

$$\mathbb{E}\Big[\sum_{j\in J} w_jC_j\Big] \;\le\; \left(1+5\delta+2\sqrt{\delta}+2\cdot\frac{\sqrt{\delta}}{1-\sqrt{\delta}}\right)\cdot Z^* \;\le\; (1+\varepsilon)\cdot Z^* \;\;.$$

The second inequality follows from the choice of $\delta=\frac{1}{\Delta}$ and a short calculation. □

4.3 The deterministic approximation scheme

Up to now we have presented a randomized approximation scheme, i. e., we can efficiently compute schedules whose *expected* values are arbitrarily close to the optimum. However, it might be more desirable to have a deterministic approximation scheme which computes schedules with a firm performance guarantee in all cases. Therefore we discuss the derandomization of the randomized approximation scheme.

Since the random variable q can only attain a constant number of different values, we can afford to derandomize the partition step of the algorithm by trying all possible assignments of values to q. In the following discussion we keep q fixed. The derandomization of the concatenation step is slightly more complicated. We use the method of conditional probabilities, i. e., we consider the random decisions one after another and always choose the most promising alternative assuming that all remaining decisions will be made randomly.

Thus, starting with the partial schedule for J_0^q, we iteratively append the remaining partial schedules for J_t^q, $t = 1, 2, \ldots$, to the current schedule. In each iteration t we use a locally optimal permutation of the machines which is given in the following way. For $i = 1, \ldots, m$, let M_i denote the load or completion time of machine i in the current schedule of the jobs in $\bigcup_{s=0}^{t-1} J_s^q$. Renumber the machines such that $M_1 \leq M_2 \leq \cdots \leq M_m$. Moreover, let W_i denote the sum of the weights of jobs in J_t^q that are processed on machine i in the corresponding partial schedule. Renumber the machines in the partial schedule such that $W_1 \geq W_2 \geq \cdots \geq W_m$. It is an easy observation, which can be proved by a simple exchange argument, that appending the partial schedule machine-wise to the current schedule according to the given numbering of machines minimizes the cost caused by the delay of jobs in J_t^q. In particular, this cost is smaller than the expected cost for the randomized variant of the algorithm which is given by

$$\sum_{s=1}^{t-1} \sum_{h \in I_s^q} L(h) \sum_{j \in J_t^q} w_j \;=\; \frac{1}{m} \sum_{i=1}^{m} M_i \sum_{i=1}^{m} W_i \;.$$

Moreover, the permutation of the machines chosen in iteration s does not influence the expected delay of jobs considered in later iterations (since the expected delay is simply the average machine load).

As a result of the above discussion, the value of the schedule computed by the deterministic algorithm is bounded from above by the expected value of the schedule computed by its randomized variant given in Subsection 4.1. Thus, as a consequence of Theorem 4.3 we can state the following main result of this paper.

Theorem 4.4. *There exists a polynomial time approximation scheme for the problem* $\mathrm{P} \,|\,|\, \sum w_j C_j$.

Acknowledgements The authors are grateful to Michel Goemans and the anonymous referees for helpful comments on the paper. Moreover, they would like to thank the organizers of the Dagstuhl-Seminar 98301 on 'Graph Algorithms and Applications' during which the result presented in this paper has been achieved.

References

[1] F. Afrati, E. Bampis, C. Chekuri, D. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, C. Stein, and M. Sviridenko. Approximation schemes for minimizing average weighted completion time with release dates. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, 1999, to appear.

[2] N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1:55–66, 1998. A preliminary version of this paper appeared in the proceedings of SODA'97.

[3] R. W. Conway, W. L. Maxwell, and L. W. Miller. *Theory of Scheduling*. Addison-Wesley, Reading, 1967.

[4] W. L. Eastman, S. Even, and I. M. Isaacs. Bounds for the optimal scheduling of n jobs on m processors. *Management Science*, 11:268–279, 1964.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP–Completeness*. Freeman, San Francisco, 1979.

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

[7] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: Off–line and on–line approximation algorithms. *Mathematics of Operations Research*, 22:513–544, 1997.

[8] H. Hoogeveen, P. Schuurman, and G. J. Woeginger. Non-approximability results for scheduling problems with minsum criteria. In R. E. Bixby, E. A. Boyd, and R. Z. Ríos-Mercado, editors, *Integer Programming and Combinatorial Optimization*, volume 1412 of *Lecture Notes in Computer Science*, 353–366. Springer, Berlin, 1998.

[9] T. Kawaguchi and S. Kyan. Worst case bound of an LRF schedule for the mean weighted flow–time problem. *SIAM Journal on Computing*, 15:1119–1129, 1986.

[10] H. Kellerer, T. Tautenhahn, and G. J. Woeginger. Approximability and nonapproximability results for minimizing total flow time on a single machine. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, 418–426, 1996.

[11] H. W. Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[12] S. Leonardi and D. Raz. Approximating total flow time on parallel machines. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, 110–119, 1997.

[13] C. Phillips, C. Stein, and J. Wein. Minimizing average completion time in the presence of release dates. *Mathematical Programming*, 82:199–223, 1998.

[14] S. Sahni. Algorithms for scheduling independent tasks. *Journal of the Association for Computing Machinery*, 23:116–127, 1976.

[15] W. E. Smith. Various optimizers for single–stage production. *Naval Research and Logistics Quarterly*, 3:59–66, 1956.

[16] M. Skutella. Semidefinite relaxations for parallel machine scheduling. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, 472–481, 1998.

[17] G. J. Woeginger. When does a dynamic programming formulation guarantee the existence of an FPTAS? In *Proceedings of the 10th Annual ACM–SIAM Symposium on Discrete Algorithms*, 820–829, 1999.