INSTITUT FÜR INFORMATIK

Scheduling unrelated parallel machines: linear programming strikes back

Klaus Jansen
Monaldo Mastrolilli

Bericht Nr. 1004
March 2010

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

ZU KIEL

Scheduling unrelated parallel machines: linear programming strikes back.

Klaus Jansen *
Institut für Informatik
Universität zu Kiel, Kiel, Germany
kj@informatik.uni-kiel.de

Monaldo Mastrolilli †
IDSIA Lugano
Galleria 2, Manno, Switzerland
monaldo@idsia.ch

March 2, 2010

Abstract

In this paper we show how to improve a recent approximation scheme for scheduling jobs on a fixed number of unrelated parallel machines. Given a set of n jobs and a set of m processors with processing times p_{ij} of job j on processor i, the goal is to find a non-preemptive schedule with minimum or approximate makespan. The previous best algorithm by Fishkin and the authors runs in time $O(n)+(\log m/\epsilon)^{O(m^2)}$. By first preprocessing the input to obtain a reduced problem instance with at most $\min\{n,(\log m/\epsilon)^{O(m)}\}$ jobs, and then by using dynamic and linear programming for different groups of jobs we are able to obtain an approximation scheme with running time $O(n)+2^{O(m\log(m/\epsilon))} \leq O(n)+(\log m/\epsilon)^{O(m\log m)}$. If ϵ is relatively small (e.g. $\epsilon < 1/m$), the running time of our scheme can be bounded by $O(n)+(1/\epsilon)^{O(m)} = O(n)+2^{O(m\log(1/\epsilon))}$.

1 Introduction

Let $\mathcal{J}$ be the set of n jobs and let $\mathcal{M}$ be the set of m machines in I, respectively. We suppose that the processing time of job j on machine i is $p_{ij} \geq 0$, for $i = 1,\ldots,m$ and $j = 1,\ldots,n$. The objective is to compute a non-preemptive schedule such that each job is executed by exactly one machine

*Research supported in part by the EU project AEOLUS contract number 015964.

†Research supported in part by the SNF Project N. 200020-122110/1"Approximation Algorithms for Machine Scheduling Through Theory and Experiments III".

and the maximum completion time among all jobs (makespan) is minimized. Lenstra, Shmoys, and Tardos [6] gave a polynomial-time approximation algorithm with ratio 2 for this problem, which currently is the best-known approximation ratio achieved in polynomial time. They also proved that there is no $1+\epsilon$ approximation algorithm for any positive $\epsilon < 1/2$, unless $P = NP$. Since the problem is NP-hard even for $m = 2$ machines [2], it is natural to ask how well the optimum can be approximated for a constant number of machines. Interestingly, Horowitz and Sahni [4] gave a fully polynomial-time approximation scheme (FPTAS) that computes an $1 + \epsilon$ approximate solution in time $O(nm(nm/\epsilon)^{m-1})$ for any $\epsilon > 0$, which is polynomial in both n and $1/\epsilon$. Lenstra et al. [6] also proposed an approximation scheme for the problem that runs in time $(n+1)^{m/\epsilon}poly(|I|)$ where $poly(|I|)$ is a polynomial of the input size. Their algorithm is not polynomial for fixed m, but has a smaller space complexity than the one by Horowitz and Sahni [4]. Jansen and Porkolab [5] presented a fully polynomial time approximation scheme for the problem whose running time is $n(m/\epsilon)^{O(m)}$. To obtain this running time, they consider long and short jobs separately, combine the previous dynamic and linear programming approaches by Horowitz and Sahni [4] and by Lenstra et al. [6], and use the price-directive decomposition method by Grigoriadis and Khachiyan [3] for computing approximate solutions of block structured linear programs and a rounding technique by Plotkin et al. [7]. Recently, Fishkin and the authors [1] found a simpler approximation scheme with running time $O(n) + (\log m/\epsilon)^{O(m^2)}$. It is based on a pre-processing step that reduces the number of jobs to $\min\{n, (\log m/\epsilon)^{O(m)}\}$ and uses a dynamic program for a constant number of jobs.

In this paper we improve the running time to $O(n)+(m/\epsilon)^{O(m)} = O(n)+2^{O(m\log(m/\epsilon))} \leq O(n) + (\log m/\epsilon)^{O(m\ln m)}$ by a careful combination of the previous algorithms in [1, 5]. We propose a two-phase approach where we first reduce the number of jobs to a number of jobs that depends on m and $1/\epsilon$ and then use a dynamic and linear program for different classes of jobs. Interestingly, the price-directive decomposition method by Grigoriadis and Khachiyan [3] to solve the underlying linear program approximately strikes here back to reduce the running time of our approximation scheme. If ϵ is small enough (e.g. $\epsilon \leq 1/m$), the running time also can be bounded by $O(n) + (1/\epsilon)^{O(m)} = O(n) + 2^{O(m\log(1/\epsilon))}$.

2 Description of the Algorithm

In the following we assume without loss of generality that $m \geq 2$ and $0 < \epsilon < 1/2$. In the first phase of our algorithm we reduce the number of jobs in

our instance I using the algorithm by Fishkin et al. [1].

2.1 Reducing the number of jobs

We round the processing times of the jobs and merge jobs of the same profile together to obtain a constant number of jobs. To do this, we first compute a lower and upper bound for the minimum objective value $OPT(I)$. Let $d_j = \min_{i=1,\ldots,m} p_{ij}$ be the minimum processing time of job j and let $D = \sum_{j=1}^{n} d_j$. It is easy to check that $OPT(I) \leq D \leq mOPT(I)$. This also implies that $OPT(I) \in [D/m, D]$. By dividing all processing times by D/m we may assume that $1 \leq OPT(I) \leq m$. For each job j we define a set of *slow machines*:

$$\mathcal{S}_j = \{i | p_{ij} \geq \frac{m}{\epsilon'} d_j\}$$

where $\epsilon' \leq 1/3$ depends on the accuracy ϵ. The input data of each job j $(j = 1, \ldots, n)$ can be rounded as follows:

(1) for any slow machine $i \in \mathcal{S}_j$ for job j, set the corresponding processing time p_{ij} to a large enough number such that no reasonable algorithm would schedule that job on a slow machine ($p_{ij} \geq 2m$ is sufficient); i.e. write $p_{ij} := \infty$.

(2) for any other machine i of job j, round the processing time p_{ij} down to the nearest lower value $d_j(1+\epsilon')^h$ for some $h \in \mathbb{N}$.

Let I_{round} be the instance with the modified and rounded processing times. We can prove the following result:

Lemma 2.1 $OPT(I_{round}) \leq (1+\epsilon')OPT(I)$.

Proof: Let us split the rounding into two phases. In the first phase we generate an instance I' by using the rounding step (2). We get $OPT(I') \leq OPT(I) \leq (1+\epsilon')OPT(I')$. To see this, consider an optimum assignment $B : \mathcal{J} \to \mathcal{M}$ of the set $\mathcal{J}$ of jobs in instance I to the set $\mathcal{M}$ of machines. Clearly, the optimum value $OPT(I')$ corresponding to B cannot be larger than $OPT(I)$ (we just reduced the processing times). By replacing the rounded processing times $\bar{p}_{ij} = d_j(1+\epsilon')^h$ by the original processing times $p_{ij} \leq \bar{p}_{ij}(1+\epsilon')$, the processing times on each machine is increased by at most the factor $(1+\epsilon')$. This implies $OPT(I) \leq (1+\epsilon')OPT(I')$.

Next, I' is transformed into I_{round} by using the rounding step (1) above. Here we obtain $OPT(I_{round}) \leq (1+\epsilon')OPT(I')$ by considering an optimum solution for I' and moving jobs j on slow machines to fastest machines. Let

$B' : \mathcal{J} \to \mathcal{M}$ be an optimum assignment of jobs to machines for instance I' and let $S \subset \mathcal{J}$ be the set of jobs scheduled on a slow machine. The effect of the modification (the additional processing time on one machine) is bounded by $\sum_{j \in S} d_j \leq \sum_{j \in S} \frac{\epsilon'}{m} p_{B'(j)j} \leq \epsilon' OPT(I') \leq \epsilon' OPT(I)$. □

Lemma 2.2 *Suppose that there is an α - approximation algorithm A_r for I_{round} with $A_r(I_{round}) \leq \alpha OPT(I_{round})$. Then, there is an approximation algorithm A for I with $A(I) \leq \alpha(1+\epsilon')^2 OPT(I)$.*

Proof: Suppose that there is an approximation algorithm A_r with ratio α for I_{round}. The approximation algorithm A' for I' works as follows: simply construct I_{round} for I' and apply A_r. This gives $A'(I') = A_r(I_{round}) \leq \alpha OPT(I_{round}) \leq \alpha(1+\epsilon')OPT(I')$. In addition the approximation algorithm A' for I' can be used also for I: here we simply construct I' for I and apply A'. After using A', we replace the modified execution times by the original processing times and obtain $A(I) \leq (1+\epsilon')A'(I') \leq \alpha(1+\epsilon')^2 OPT(I') \leq \alpha(1+\epsilon')^2 OPT(I)$. □

In a similar way as above we can prove the following result.

Lemma 2.3 *Suppose that there is an approximation algorithm A_r for I_{round} with $A_r(I_{round}) \leq \alpha OPT(I_{round}) + \beta$. Then there exists an approximation A for I with $A(I) \leq \alpha(1+\epsilon')^2 OPT(I) + \beta(1+\epsilon') \leq [\alpha(1+\epsilon')^2 + \beta(1+\epsilon')]OPT(I)$.*

Consider now the instance I_{round}. The *execution profile* of a job j is an m-tuple $(\Pi_{1,j}, \ldots, \Pi_{m,j})$ such that $p_{ij} = d_j(1+\epsilon')^{\Pi_{ij}}$ where we use the convention that $\Pi_{i,j} = \infty$ if $p_{ij} = \infty$. Two jobs have the same profile, if and only if they have the same execution profile. The number of distinct profiles is at most $\ell = (2 + 2\log_{1+\epsilon'} \frac{m}{\epsilon'})^m$. Let $\gamma = 1/\lceil m/\epsilon' \rceil$. Then we partition the set of jobs into two subsets $L = \{j | d_j > \gamma\}$ and $S = \{j | d_j \leq \gamma\}$. L is the set of *large jobs* and S the set of *small jobs*. The small jobs are further partitioned into subsets S_σ of jobs having the same profile, for $\sigma = 1, \ldots, \ell$. Two jobs j_a and j_b from the same set S_σ with $d_{j_a}, d_{j_b} \leq \gamma/2$ are grouped together to a composed job j_c in which the processing time on machine i is equal to the sum of the processing times of j_a and j_b on machine i, and set $d_{j_c} = d_{j_a} + d_{j_b}$. This process is repeated until at most one job $j \in S_\sigma$ has $d_j \leq \gamma/2$. All other jobs have processing times larger than $\gamma/2$. The number of jobs in the modified instance I_{merge} is bounded by $\frac{2D}{\gamma} + \ell \leq \frac{2m(m+\epsilon')}{\epsilon'} + \ell = (\log m/\epsilon)^{O(m)}$ using $\gamma \geq \frac{\epsilon'}{m+\epsilon'}$, $D = m$ after diving all processing times by D/m and $\epsilon' = \Theta(\epsilon)$. Using these observations we obtain:

Lemma 2.4 *The number of jobs in I_{merge} is at most* $\min\{n, (\log m/\epsilon)^{O(m)}\}$ *and the objective value $OPT(I_{round}) \leq OPT(I_{merge}) \leq OPT(I_{round}) + \epsilon'$.*

Proof: Consider the instance I_{round}. Clearly, $OPT(I_{round}) \leq OPT(I_{merge})$ (since merged jobs have less freedom in the possible assignments). In the following we show that there is a schedule for I_{merge} of length at most $OPT(I_{round}) + \epsilon'$. Consider an optimum solution $B : \mathcal{J} \to \mathcal{M}$ for I_{round} of value $OPT(I_{round})$. Suppose that each machine executes the large jobs in I_{round} at the beginning of the schedule. Let $t_i \geq 0$ be the time at which machine i finishes to process large jobs for $i = 1, \ldots, m$. Consider the following linear program LP that is a relaxation of the original integer linear program ILP for the small jobs in I_{round} (after placing the large jobs as in the optimum solution):

$$\begin{array}{lll} Min & T & \\ s.t. & \sum_{j \in S} x_{ij} d_j (1+\epsilon')^{\Pi_{ij}} + t_i \leq T & i = 1, \ldots, m \\ & \sum_{i=1}^{m} x_{ij} = 1 & j \in S \\ & x_{ij} \geq 0, i = 1, \ldots, m, & j \in S. \end{array}$$

Therefore, the minimum objective value $OPT(LP) \leq OPT(I_{round})$. For each subset S_σ, $\sigma = 1, \ldots, \ell$, we use a set of variables $y_{i\sigma} \in [0,1]$ for $i = 1, \ldots, m$ to represent the fraction of jobs from S_σ processed on machine i. The following linear program LP'

$$\begin{array}{lll} Min & T & \\ s.t. & \sum_{\sigma=1}^{\ell} y_{i\sigma} \sum_{j \in S_\sigma} d_j (1+\epsilon')^{\Pi_{ij}} + t_i \leq T & i = 1, \ldots, m \\ & \sum_{i=1}^{m} y_{i\sigma} = 1 & \sigma = 1, \ldots, \ell \\ & y_{i\sigma} \geq 0, i = 1, \ldots, m, & \sigma = 1, \ldots, \ell \end{array}$$

is also a relaxation of the ILP with the same objective value $OPT(LP) = OPT(LP')$. To see this set

$$y_{i\sigma} = \frac{\sum_{j \in S_\sigma} x_{ij} d_j (1+\epsilon)^{\Pi_{i,j}}}{\sum_{j \in S_\sigma} d_j (1+\epsilon)^{\Pi_{i,j}}}.$$

Then any feasible solution (x_{ij}) for LP gives also a feasible solution $(y_{i\sigma})$ for LP'. Furthermore,

$$\sum_{\sigma=1}^{\ell} y_{i\sigma} \sum_{j \in S_\sigma} d_j (1+\epsilon)^{\Pi_{i,j}} = \sum_{\sigma=1}^{\ell} \sum_{j \in S_\sigma} x_{ij} d_j (1+\epsilon)^{\Pi_{i,j}} = \sum_{j \in S} x_{ij} d_j (1+\epsilon)^{\Pi_{i,j}}$$

for any $i = 1, \ldots, m$. This implies that the objective value $OPT(LP')$ is equal to $OPT(LP)$.

Now we prove that there is a schedule of length $OPT(I_{round}) + \epsilon'$ if we use the grouped small jobs in I_{merge}. Let us take an optimum solution $(y^*_{i\sigma})$

for LP'. For every $y^*_{i\sigma} > 0$ schedule a subset of grouped jobs from S_σ on machine i until either all jobs from S_σ are placed or the total fraction of jobs assigned to i is equal to $y^*_{i\sigma}$. If necessary, we preempt one job to use the fraction $y^*_{i\sigma}$ completely. If $y^*_{i\sigma}$ is integral for $i = 1, \ldots, m$ then all jobs from S_σ are not preempted. The total number of preempted jobs from S_σ is at most $f_\sigma - 1$ where $f_\sigma = |\{i | y_{i\sigma} > 0, i = 1, \ldots, m\}|$. All preempted jobs j are removed and scheduled at the end of the schedule on machine m_j where $p_{m_j j} = d_j$. This increases the makespan by at most $\Delta = \gamma \sum_{\sigma=1}^{\ell}(f_\sigma - 1)$, since the processing time of each grouped small job j is most $d_j \leq \gamma$ when executed on the fastest machine m_j. The number of strictly positive variables in any basic solution of LP' is at most the number m of machines [6]. Therefore, $\sum_{\sigma=1}^{\ell}(f_\sigma - 1) \leq m$. To bound the total increase $\Delta \leq \gamma m$ for the makespan by ϵ', we use $\gamma \leq \epsilon'/m$. Notice that $\gamma \leq \epsilon'/m$ and that $1/\gamma$ is integral. □

Notice that an α approximation algorithm A_m for I_{merge} implies an approximation algorithm A_r for I_{round} with $A_r(I_{round}) \leq \alpha OPT(I_{round}) + \alpha\epsilon'$. Therefore, this gives also an approximation algorithm A for the original instance I with $A(I) \leq \alpha(1+\epsilon')^2 OPT(I) + \alpha\epsilon'(1+\epsilon') \leq [\alpha(1+\epsilon')^2 + \alpha\epsilon'(1+\epsilon')]OPT(I) \leq [\alpha(1+3\epsilon'+2(\epsilon')^2)]OPT(I)$ using $OPT(I) \geq 1$.

In the second phase of our algorithm we use the algorithm by Jansen and Porkolab [5] for the transformed instance. Suppose that I_{merge} contains $n' \leq min\{n, (\log m/\epsilon)^{O(m)}\}$ jobs $1, \ldots, n'$ with processing times p'_{ij}. First we compute again the $d'_j = \min_i p'_{ij}$ values for the n' jobs in I_{merge} and set $D' = \sum_j d'_j$. Clearly, $OPT(I_{merge}) \leq D' \leq mOPT(I_{merge})$. We select now $K = m\lceil m/\epsilon' \rceil$ longest jobs with respect to the d'_j values. Then we find the smallest $k \leq K - m$ such that $d'_{k+1} + \ldots + d'_{k+m} \leq (\epsilon'/m)D' \leq \epsilon' OPT(I_{merge})$ and partition the set of jobs into two sets T_ℓ and T_s such that T_ℓ contains the k longest jobs according to the d'_j. Selecting the K longest jobs and finding the smallest k with the above property can be done in $O(Kn') = O(m^2 n'/\epsilon')$.

2.2 Dynamic programming

Via dynamic programming we construct the set $T(k)$ of all substantially different (up to $\delta = \epsilon'/4$ accuracy) schedules for jobs in T_ℓ. To do this we divide the interval $[0, D']$ where $D' \leq mOPT(I_{merge})$ into $N = mk/\delta$ equal parts $S_1, \ldots, S_N$ each of size $D'\delta/mk$. Let $T(j)$ denote a set of m-tuples consisting of the m completion times for different schedules of the first j jobs $1, \ldots, j$. The algorithm iteratively computes the sets $T(j)$ starting with $T(0) = \{(0, \ldots, 0)\}$. Implicitly we round up the processing time of each job to a multiple of $D'\delta/mk$, i.e. $\tilde{p}_{ij} = \alpha_{ij} D'\delta/mk \leq p'_{ij} + D'\delta/mk$ where $\alpha_{ij} \in \{1, \ldots, mk/\delta\}$. Then we work with the rounded processing times and construct partial schedules by adding a processing time $\tilde{p}_{ij}$ for each machine

i to each vector $t \in T(j-1)$ and identifying those vectors with the same finishing times. At each step when it proceeds from $(j-1)$ to j, the error it makes can be bounded by the length of the intervals or the rounding factor, i.e. by $D'\delta/mk$. Thus the total error is bounded by $kD'\delta/mk = D'\delta/m \leq \delta OPT(I_{merge})$. Since the number of vectors is at most $N^m = (mk/\delta)^m$, the dynamic program runs in time $O(kmN^m) = (km/\delta)^{O(m)}$.

Lemma 2.5 *Our dynamic program computes a set $T(k)$ of m-tuples $(t_1, \ldots, t_k)$ in time $(km/\delta)^{O(m)}$ of size at most $(mk/\delta)^m$. For each schedule of the jobs with machine completion times $(f_1, \ldots, f_m)$, there is a tuple $(t_1, \ldots, t_m) \in T(k)$ such that $t_i \leq f_i + \delta OPT(I_{merge})$.*

2.3 Linear Programming

For every m-tuple $t = (t_1, \ldots, t_m) \in T(k)$ consider the following integer linear program $ILP(t)$:

$$\begin{array}{lll} Min & \lambda & \\ s.t. & \sum_{j=k+1}^{n'} p'_{ij} x_{ij} + t_i \leq \lambda & i = 1, \ldots, m \\ & \sum_{i=1}^{m} x_{ij} = 1 & j = k+1, \ldots, n' \\ & x_{ij} \in \{0,1\}, i = 1, \ldots, m, & j = k+1, \ldots, n'. \end{array}$$

It describes the original scheduling problem for the set of short jobs T_s assuming that machine i can start processing jobs only at time t_i (due to the schedule of long jobs corresponding to t). In any feasible solution of $ILP(t)$, $x_{ij} = 1$ indicates that j is scheduled on machine i. Let $LP(t)$ denote the linear programming relaxation of $ILP(t)$, which is obtained by replacing every $0-1$ constraint $(x_{ij} \in \{0,1\})$ with the weaker nonnegativity condition $(x_{ij} \geq 0)$. $LP(t)$ can be interpreted as a block optimization problem, where the blocks are m-dimensional simplicies $B_j = \{x_j \in \mathbb{R}^m | \sum_{i=1}^{m} x_{ij} = 1, x_{ij} \geq 0, i = 1, \ldots, m\}$, for $j = k+1, \ldots, n$, and the coupling constraints are the linear inequalities $\sum_{j=k+1}^{n'} p'_{ij} x_{ij} + t_i \leq \lambda$, for $i = 1, \ldots, m$, corresponding to the m machines. By using the Logarithmic Potential Price Directive Decomposition Method developed by Grigoriadis and Khachiyan [3], we can compute a ρ-approximate solution $x(t)$ of $LP(t)$ in $O(m(\ln m + \rho^{-2} \ln \rho^{-1}))$ iterations, where each iteration requires $O(m \ln \ln(m\rho^{-1}))$ operations and $(n' - k)$ block optimizations performed to a relative accuracy $\rho/6$. Each block optimization here is the minimization of a given linear function over an m-dimensional simplex. This can be done here optimally (not only approximately) in $O(m)$ time. Therefore, we obtain an approximate solution of $LP(t)$ in $O(n'(m/\rho)^2 \ln(m/\rho) \ln \ln(m/\rho))$. Furthermore, the LP solution $x(t)$ can be converted in $O(|E|m) = O(nm^2)$ time into another fractional

assignment of the same length represented by a forest [7]. The number of jobs in such a fractional assignment with non-unique job assignments can be bounded by $m-1$. This implies the following result.

Lemma 2.6 *For any $t \in T(k)$, a ρ-approximate solution $x(t)$ of $LP(t)$ can be computed in $O(n'(m/\rho)^2 \ln(m/\rho) \ln\ln(m/\rho))$ time such that $x(t)$ provides a fractional assignment for less than m jobs.*

We choose a fractional assignment $x(t)$ among all $t \in T(k)$ with the smallest objective value $\lambda(x(t)) = \max\{\sum_{j=k+1}^{n'} p_{ij}x(t)_{ij} + t_i | i = 1, \ldots, m\}$. Assume that it is attained for $t^* \in T(k)$ and let $x(t^*)$ be the assignment vector corresponding to t^*.

Lemma 2.7 *The length of the schedule by the vector t^* for the long jobs and the assignment vector x^* for the small jobs with unique machine-job values is bounded by $(1+\delta)(1+\rho)OPT(I_{merge})$.*

Let $\{i_1, \ldots, i_p\}$ be the set of jobs with non-unique machine assignment in $x(t^*)$. Since there are at most m jobs with fractional assignments, the selection of k implies $d'_{i_1} + \ldots + d'_{i_m} \leq \epsilon' OPT(I_{merge})$. Executing these jobs at the end sequentially according to the d'_i values (i.e. each of them is assigned to the machine where its execution time is the smallest) gives a total length of at most $\epsilon' OPT(I_{merge})$. Let $\bar{x}$ be the assignment vector of the complete schedule. Combining with the bound above the schedule length $S(\bar{x})$ is at most $[(1+\delta)(1+\rho)+\epsilon']OPT(I_{merge})$. This implies an approximation algorithm for I with schedule length at most $[((1+\delta)(1+\rho)+\epsilon')(1+3\epsilon'+2(\epsilon')^2)]OPT(I)$. To guarantee that the makespan of the computed schedule is at most $(1+\epsilon)OPT(I)$, we have to choose ρ, δ and ϵ' appropriately. For example, $\delta = \rho = \epsilon'$ and $\epsilon' = \epsilon/8$ suffices. All steps in the second phase of our algorithm require at most $n'(m/\epsilon)^{O(m)}$ time and $n' \leq \min\{n, (\log m/\epsilon)^{O(m)}\}$. Therefore, the overall running time can be bounded by $O(n) + (m/\epsilon)^{O(m)} = O(n) + 2^{O(m\log(m/\epsilon))}$. Since $m \leq (\log m)^{\log m} + 1$, $(m/\epsilon) \leq 2(\log m)^{\log m}/\epsilon \leq 2(\log m/\epsilon)^{\log m}$ for $m \geq 2$. This implies that $2^{\log(m/\epsilon)} \leq 2 \cdot 2^{\log m \log(\log m/\epsilon)} = 2 \cdot (\log m/\epsilon)^{\log m}$. Therefore, the running time is also bounded by $O(n) + (\log m/\epsilon)^{O(m\log m)}$. This proves our main result.

Theorem 2.1 *There is an ϵ-approximation scheme for the non-preemptive minimum makespan problem with n jobs and m unrelated parallel machines that runs in*

$$O(n) + (m/\epsilon)^{O(m)} = O(n) + 2^{O(m\log(m/\epsilon))} = O(n) + (\log m/\epsilon)^{O(m\log m)}.$$

The space required in our algorithm is dominated by the dynamic program and is bounded by $(m/\epsilon)^{O(m)} = 2^{O(m \log(m/\epsilon))} = (\log m/\epsilon)^{O(m \log m)}$. If the accuracy ϵ is relatively small, e.g. $\epsilon < 1/m$, then the running time can be bounded by $O(n) + (1/\epsilon)^{O(m)}$ and the space complexity by $(1/\epsilon)^{O(m)}$.

3 Conclusion

In the paper we have presented an improved approximation scheme for scheduling jobs on parallel machines. Comparing to the previous result, it reduces the exponent $O(m^2)$ to $O(m \log m)$. The algorithm described above can be generalized also for scheduling on unrelated machines with costs [8]. We obtain by combining the algorithms in [1, 5] an approximation algorithm that for any $\epsilon > 0$ computes in $O(n) + 2^{O(m \log(m/\epsilon))} = O(n) + (\log m/\epsilon)^{O(m \log m)}$ time a schedule with cost at most $(1+\epsilon)C$ and makespan at most $(1+\epsilon)T$, if one of cost C and makespan T exists. Furthermore, we obtain an ϵ-approximation scheme of running time $O(n) + 2^{O(m \log(m/\epsilon))} = O(n) + (\log m/\epsilon)^{O(m \log m)}$ for minimizing the weighted sum $T + \kappa C$ of the makespan T and the cost C in non-preemptive scheduling of jobs on unrelated parallel machines, where $\kappa > 0$. Similar to the results above, the running times can be bounded by $O(n) + (1/\epsilon)^{O(m)}$ if ϵ is relatively small, e.g. $\epsilon \leq 1/m$. Interesting research directions for the future are approximation scheme for small numbers m of machines and for accuracies $\epsilon \in [1/m, 1]$.

References

[1] A. Fishkin, K. Jansen, and M. Mastrolilli: Grouping techniques for scheduling problems: simpler and faster, *Algorithmica*, 51 (2008), 183-199.

[2] M.R. Garey and D.S. Johnson: *Computers and Intractability: a Guide to the Theory of NP-completeness*, W.H. Freeman, 1979.

[3] M.D. Grigoriadis and L.G. Khachiyan: Coordination complexity of parallel price-directive decomposition, *Mathematics of Operations Research*, 21 (1996), 321-340.

[4] E. Horowitz and S. Sahni: Exact and approximate algorithms for scheduling nonidentical processors, *Journal of the ACM*, 23 (1976), 317-327.

[5] K. Jansen and L. Porkolab: Improved approximation schemes for scheduling unrelated parallel machines, *Mathematics of Operations Research*, 26 (2001), 324-338.

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

[7] S.A. Plotkin, D.B. Shmoys, and E. Tardos: Fast approximation algorithms for fractional packing and covering problems, *Mathematics of Operations Research*, 20 (1995), 257-301.

[8] D.B. Shmoys and E. Tardos: An approximation algorithm for the generalized assignment problem, *Mathematical programming*, Series A, 62 (1993), 461-474.