The Sixth International Symposium on Operations Research and Its Applications (ISORA'06)
Xinjiang, China, August 8–12, 2006
, pp. 258–263

Scheduling with Discretely Compressible Processing Times to Minimize Makespan

Shu-Xia Zhang[1,2,*] Zhi-Gang Cao[3] Yu-Zhong Zhang[3]

[1] Department of Mathematics, East China Normal University, Shanghai 200062, China
[2] Department of Watercraft Command，Zhenjiang Watercraft College, Zhenjiang, Jiangsu 212003, China
[3] College of Operations Research and Management Science, Qufu Normal University, Rizhao, Shandong 276826, China

Abstract In the classical scheduling problems, it is always assumed that the processing time of a job is fixed. In this paper, we address the scheduling with discretely compressible processing times, i.e., the possible processing time of a job can only have finitely many possibilities. Of course, choosing to process any job with a compressed processing time incurs a compression cost. We consider the single machine scheduling problem with discretely compressible processing times. The objective is to minimize makespan with the constraint of total compression cost, i.e., the total compression cost is at most a pre-given constant number. Jobs may have different release times. We also consider the identical-parallel-machine version with simultaneous release times. Since they are all NP-hard, we design for the first time pseudo-polynomial time algorithms and FPTASs. For the first problem, our main approach is dynamic programming and scaling-and-rounding; and for the second one, our approach is dynamic programming and geometry partitioning.

Keywords scheduling; discretely compressible processing times; dynamic programming; FPTAS; makespan

1 Introduction

Scheduling problems with controllable processing times have their deep root in the real world and attracted increasing attention recently. In the classical scheduling, it is always assumed that the parameters of a job, e.g. its processing time and its release time, are all fixed. However, in the real world, the processing of jobs is determined not only by the machine speed, but also by other resources such as labor, funds etc., then the parameters of a job may be not fixed. We can compress the original processing time. Of course, choosing to process any job with a compressed processing time incurs a compression cost.

Let $\{J_1, J_2, \cdots, J_n\}$ denote a list of given jobs. We write SCP as an abbreviation of the scheduling problem with compressible processing times. We denote by

*Corresponding author. Tel: 13561954187. E-mail: 52040601019@student.ecnu.edu.cn

TPC the total processing cost in SCP. There are two variants for SCP, the continuous one and the discrete one, which are denoted by SCCP and SDCP, respectively. In SCCP, any job J_j can be processed with a processing time $p_j \in [l_j, u_j]$ and incurs a corresponding compression cost $c_j(u_j - p_j)$, where c_j is the cost coefficient. And in SDCP, p_j can choose a value p_{ji} from among $\{p_{j1}, p_{j2}, \cdots, p_{jk}\}$, and the corresponding compression cost is e_{ji}, where $1 \leq i \leq k$. We assume that $p_{j1} < p_{j2} < \cdots < p_{jk}$ and $e_{j1} > e_{j2} > \cdots > e_{jk}$. This assumption is sound as the more processing time we compress the more cost we should pay.

There are the following four models for SCP:

(P1) To minimize $F_1 + F_2$;

(P2) To minimize F_1 subject to $F_2 \leq a$;

(P3) To minimize F_2 subject to $F_1 \leq b$;

(P4) To identify the set of Pareto-optimal points for (F_1, F_2).

Where F_1 is the original objective function, and F_2 is TPC. In the objective function field of the notation of Graham et al.([1]), we write the above four model as $F_1 + F_2$, F_1/F_2, F_2/F_1 and (F_1, F_2), respectively. Following Chen et al.([2]), we also use *cm* and *dm* to characterize SCCP and SDCP, respectively.

There have been many results for SCCP (for a survey till 1998, see [3]), but only four papers discussed SDCP up to now, to the best of our knowledge. As to the P1 model, Vickson([4]) showed that the P1 model for $1|dm|T_{\max}$ is NP-hard. Deniels and Mazzola([5]) studied an NP-hard flow shop scheduling problem in which the processing time of each job can be varied according to the allocation of a limited amount of resource. Chen, Lu and Tang([2]) solved the P1 models for $1|dm|\sum C_j$ and $1|d_j = D, dm|\alpha\sum E_j + \beta\sum T_j$ by formulating them as assignment problems, where D is a large enough number. They also showed that the P1 models for $1|r_j, dm|C_{\max}$, $1|d_j \equiv d, dm|T_{\max}$ and $1|d_j \equiv d, dm|w\sum U_j$ are all NP-hard and designed pseudo-polynomial time algorithms for $1|r_j, dm|C_{\max} + TPC$, $1|dm|T_{\max} + TPC$ and $1|dm|\sum w_j U_j$. Chen, Potts and Woeginger([3]) summarized these results. Cao, Wang, Zhang, and Liu([6]) showed that the P2 model for $1|dm|C_{\max}$ is NP-hard and designed an FPTAS(Fully Polynomial Time Approximation Scheme) for it.

In this paper, we address for the first time the P2 models for $1|r_j, dm|C_{\max}$ and $P_m|dm|C_{\max}$. Since they are all NP-hard, we design FPTASs for them. The rest of this paper is organized as follows. Some preliminaries are given in Section 2. In section 3, we design dynamic programming algorithm and FPTAS for $1|r_j, dm|C_{\max}/TPC$. In Section 4, we design dynamic programming algorithm and FPTAS for $P_m|dm|C_{\max}/TPC$. Conclusion and remarks are given in Section 5.

2 Preliminaries

An algorithm A is a $(1+\varepsilon)$-approximation algorithm for a minimization problem if it produces a solution which is at most $(1+\varepsilon)$ times the optimal one. A family of algorithms $\{A_\varepsilon\}_\varepsilon$ is called a polynomial time approximation scheme if, for every

$\varepsilon > 0$, the algorithm $\{A_\varepsilon\}$ is a $(1+\varepsilon)$-approximation algorithm running in time polynomial in the input size when ε is treated as constant. It is called a fully polynomial time approximation scheme if the running time is also polynomial in $1/\varepsilon$.

Let the jobs be indexed so that $r_1 \leq r_2 \leq \cdots \leq r_n$. For convenience, we assume(without loss of generality)that $r_1 = 0$. For each job $J_j, 1 \leq j \leq n$, we assume $p_{j1} > p_{j2} > \cdots > p_{jk}, e_{j1} < e_{j2} < \cdots < e_{jk}$.

Lemma 1. *([6])* $1|dm|C_{\max}/TPC$ *is NP-hard.*

Proof. Since $1|r_j,dm|C_{\max}/TPC$ and $P_m|dm|C_{\max}/TPC$ both take $1|dm|C_{\max}/TPC$ as a special case, they are NP-hard. □

3 Single machine problem with nonidentical release times

Given a set of jobs $\{J_j = (r_j; p_{j1}, p_{j2}, \cdots, p_{jk}; e_{j1}, e_{j2}, \cdots, e_{jk}) : 1 \leq j \leq n\}$ and a threshold E, we will find a schedule with the minimum makespan whose TPC is at most E.

For any partial schedule for jobs $\{J_1, \cdots, J_j\}$, if its makespan is P, we say that its state is (j,P). If (j,P) can be obtained by some partial schedule, we say it's feasible. Let S_j denote the jth state space, i.e., the set of all the feasible states obtained by partial schedules for jobs $\{J_1, \cdots, J_j\}$. For any $(j,P) \in S_j$, $M(j,P)$ represents the minimum TPC of partial schedules whose states are (j,P).

We define $r_{\max} = \max\{r_j : 1 \leq j \leq n\}$, $P_{\text{sum}} = \sum_{j=1}^{n} p_{j1}$. Based on the preprocessing proceeds proprosed by C.K. Poon and P. Zhang([7]), for $1|r_j,dm|C_{\max}/TPC$, if $r_{\max} \geq P_{\text{sum}}$, an optimal sequence can be obtained easily in $O(n \log n)$ time. So we assume that $r_{\max} < P_{\text{sum}}$ in the following.

Lemma 2. *The smallest-release-time first order is an optimal sequence for* $1|r_j,dm|C_{\max}/TPC$.

The result is straightforward and we thus omit the proof. So we design in the following a dynamic programming algorithm which iteratively constructs a schedule by assigning an unscheduled job with the smallest release time to the last position of the current schedule.

Algorithm OMM (*Optimal Minimum Makespan*)

As to initialization, we let $S_0 = \{(0,0),(0,1),\cdots,(0,r_{\max})\}$ and $M(0,P) = 0$, where $0 \leq P \leq r_{\max}$.

In stage j, we first let $S_j = \emptyset$ and then for each $(j-1,P) \in S_{j-1}$, if $P \geq r_j$, we add $(j,P+p_{j1})$, $(j,P+p_{j2})$, $\cdots$, $(j,P+p_{jk})$ to S_j. After S_j is constructed, for any $(j,P) \in S_j$, let $M(j,P) = \min\{M(j-1,P-p_{ji}) + e_{ji} : 1 \leq i \leq k\}$.

In order to get the optimal schedule, we only have to find the minimum P such that $M(n,P) \leq E$ and derive a corresponding schedule whose state is (n,P) by backtracking.

It's straightforward that the time complexity is $O(nk(r_{\max}+P_{\text{sum}}))$, since $r_{\max} < P_{\text{sum}}$, the time complexity is $O(nkP_{\text{sum}})$, which is pseudo-polynomial.

Algorithm OMM solves the $1|r_j,dm|C_{\max}/TPC$ problem optimally, we now extend this algorithm to an FPTAS.

We reindex all the possible processing times such that: $p^1 \geq p^2 \geq \cdots \geq p^{nk}$, where $p^t = p_{j_t i_t}, 1 \leq j_t \leq n, 1 \leq i_t \leq k$. We define: $t_0 = arg\{p^{t_0} = \max\{p_{1k}, p_{2k}, \cdots, p_{nk}\}\}$. We denote $\mathbf{OMM}_i$ as the dynamic programming algorithm **OMM** which schedules jobs without processing times $\{p^1, p^2, \cdots, p^i\}$. We also denote **OMM** as algorithm $\mathbf{OMM}_0$.

Algorithm AMM(ε) (*Approximate Minimum Makespan*)

Input an instance $\mathscr{J}$.

For $s = 0$ **to** $t_0 - 1$ **do**

Let $M_s := \max\{r_{\max}, p^{s+1}\}$, $M_s^{'} := \varepsilon M_s/(n+1)$;

Construct a rounded down instance $\tilde{\mathscr{J}}_j = \{(r_j^{'}; p_{j1}^{'}, \cdots, p_{jk}^{'}; e_{j1}, \cdots, e_{jk}) : 1 \leq j \leq n\}$, where $r_j^{'} = M_s^{'}\lfloor r_j/M_s^{'}\rfloor, p_j^{'} = M_s^{'}\lfloor p_j/M_s^{'}\rfloor$(Take $M_s^{'}$ as a unit in the rounded down instance, i.e., scale it);

Call algorithm OMM_s to find an optimal schedule $\tilde{\pi}_s$ for $\tilde{\mathscr{J}}$;

If $\tilde{\pi}_s$ exists **then** obtain the corresponding schedule π_s for $\mathscr{J}$ from $\tilde{\pi}_s$

else $h := s$, break;

EndFor

If $h = 0$ **then** output "no feasible schedule"

else find π_{i^0} with the minimum makespan from $\{\pi_s : 0 \leq s \leq h-1\}$.

Output π_{i^0} as the solution to the original instance $\mathscr{J}$.

Given a schedule π, we denote $C_{max}(\pi)$ as the makespan of π. Let $\pi^* = (p_{1x_1}, p_{2x_2}, \cdots, p_{nx_n})$ be the optimal schedule for the original problem. We suppose that $p_{lx_l} = \max\{p_{1x_1}, p_{2x_2}, \cdots, p_{nx_n}\}$, and $p^{t_1} = p_{lx_l}$, then $p^{t^1} \geq p^h$, hence $t^1 \leq h$, i.e., $t_1 - 1 \leq h-1$. We denote $\tilde{\pi}^*$ as the corresponding schedule of π^* for the rounded down instance. Obviously $\tilde{\pi}^*$ is a feasible schedule for $\tilde{\mathscr{J}}_{t_1-1}$. We will prove: $C_{\max}(\pi_{t_1-1}) \leq (1+\varepsilon)C_{\max}(\pi^*)$.

Due to rounding, the start time of the last block in π_{t_1-1} may be later than that of the corresponding last block in $\tilde{\pi}_{t_1-1}$, and its processing time may also be longer, but at most longer by $M_{t_1-1}^{'}$(in the original scale). So we have $C_{\max}(\pi_{t_1-1}) \leq C_{\max}(\tilde{\pi}_{t_1-1}) + (n+1)M_{t_1-1}^{'} \leq C_{\max}(\tilde{\pi}^*) + \varepsilon\max\{r_{\max}, p^{t_1}\} \leq C_{\max}(\pi^*) + \varepsilon C_{\max}(\pi^*) = (1+\varepsilon)C_{\max}(\pi^*)$. Since $C_{\max}(\pi_{i_0}) \leq C_{\max}(\pi_{t_1-1})$, we have $C_{\max}(\pi_{i_0}) \leq (1+\varepsilon)C_{\max}(\pi^*)$.

Now, let's calculate the running time. As the main time of algorithm AMM(ε) is calling algorithm OMM, OMM_1,$\cdots$,OMM_{h-1} in every step, its running time is: $O\left((h-1)nk\left(r_{\max}^{'} + P_{\text{sum}}^{'}\right)\right) \leq O\left(n^2k^2P_{\text{sum}}^{'}\right) \leq O\left(n^2k^2nP_{\max}^{'}\right) = O\left(n^3k^2\left\lfloor P_{\max}/M_0^{'}\right\rfloor\right) = O\left(n^4k^2\left(\frac{1}{\varepsilon}\right)\right)$

It is polynomial both in n and $\frac{1}{\varepsilon}$. According to the above analysis, we have:

Theorem 3. *$1|r_j,dm|C_{\max}/TPC$ admits an FPTAS.*

4 Parallel machine problem with identical release times

Given a set of jobs $\{J_j = (p_{j1}, p_{j2}, \cdots, p_{jk}, e_{j1}, e_{j2}, \cdots, e_{jk}) : 1 \leq j \leq n\}$, a set of identical parallel machines$\{M_1, M_2, \cdots, M_m\}$ and a threshold H, where m is a constant, each job has to be processed by exactly one machine, we will find a schedule with the minimum makespan whose TPC is at most H.

For any partial schedule for jobs $\{J_1, \cdots, J_j\}$, if the load on machine M_i is c_i, let $C = (c_1, c_2, \cdots, c_m)^T$, which is an $m-$dimensional vector, we say that its state is (j, C). If (j, C) can be obtained by some partial schedule, we say it's feasible. Let S_j denote the jth state space, i.e., the set of all the feasible states obtained by partial schedules for jobs $\{J_1, \cdots, J_j\}$. For any $(j, C) \in S_j$, $M(j, C)$ represents the minimum TPC of partial schedules whose states are (j, C).

In stage j, we first let $S_j = \emptyset$. For each $(j-1, C) \in S_{j-1}$, $1 \leq u \leq k$ and $1 \leq v \leq m$, we add $(j, C + p_{ju}E_v)$ to S_j, where E_v is an $m-$dimensional unit vector whose vth component is 1. After S_j is constructed, for each $(j, C) \in S_j$, let $M(j, C) = \min\{M(j-1, C - p_{ji}E_v) + e_{ji} : 1 \leq i \leq k\}$.

As to initialization, $S_1 = \{(1, p_{1u}E_v) : 1 \leq u \leq k, 1 \leq v \leq m\}$ and $M(1, p_{1u}E_v) = e_{1u}$. And to get the optimal schedule, we only have to find the minimum makespan and corresponding C, such that $M(n, C) \leq H$ and derive a corresponding schedule whose state is (n, C) by backtracking.

It's straightforward that the time complexity is $O(nk(nP_{\max})^m)$, which is pseudo-polynomial.

To further get an FPTAS, we have to trim the state space S_j. We denote T_j as the new state space trimmed down. Given any accuracy parameter $\varepsilon > 0$, let $\varepsilon_0 = \varepsilon/(2n)$. We partition the time horizon $[0, nP_{\max}]$ into intervals $I_0 = [0, 1], I_1 = ((1+\varepsilon_0)^0, (1+\varepsilon_0)^1], I_2 = ((1+\varepsilon_0)^1, (1+\varepsilon_0)^2], \cdots, I_t = ((1+\varepsilon_0)^{t-1}, (1+\varepsilon_0)^t]$, where $t = \lceil \log_{1+\varepsilon_0}^{nP_{max}} \rceil$.

We initialize $T_0 = \{(1, p_{1u}E_v) : 1 \leq u \leq k, 1 \leq v \leq m\}$. In the jth stage, $1 \leq j \leq n$, we first compute S_j from T_{j-1} as we do in the original dynamic programming. For any $(j, C) \in S_j$, if $c_t (1 \leq t \leq m)$ falls into the ith time interval I_i, $i \geq 1$, we let $c_t^{'} = (1+\varepsilon_0)^i$; and if $c_t = 0$ or $c_t = 1$, we simply let $c_t^{'} = c_t$. Then we add the new state $(j, C^{'})$ to T_j and let $M^{'}(j, C^{'}) = M(j, C)$, where $C^{'} = (c_1^{'}, c_2^{'}, \cdots, c_m^{'})^T$. Notice that $c_t^{'}$ is at most $(1+\varepsilon_0)$ times of c_t and the cardinality of T_j for any $1 \leq j \leq n$ is at most $(\lceil \log_{1+\varepsilon_0}^{nP_{\max}} \rceil)^m$, which is a polynomial in the input size. We find the minimum makespan and corresponding $C^{'}$ such that $M^{'}(j, C^{'}) \leq H$. Suppose that π is a corresponding schedule, We will verify that π meets our demand. Notice first that π is a feasible schedule.

Theorem 4. *The makespan of π is at most $(1+\varepsilon)$ times that of the optimal one.*

Proof. π assigns the actual processing time p_{ja_j} of job $J_j (1 \leq j \leq n)$ to some machine. Obviously, the load on each machine may not be $c_i^{'}$ but a little smaller. Let π^* be the optimal schedule and $C^* = (c_1^*, c_2^*, \cdots, c_m^*)^T$. Its corresponding state in T_n is $(n, C^{*'})$, where $C^{*'} = (c_1^{*'}, c_2^{*'}, \cdots, c_n^{*'})^T$. We denote $p_{i_1 o_{i1}}^*$ and $p_{i_s o_{is}}^*$ as the first and the last actual processing time on machine M_i in π^*, respectively. The

load c_i^* on machine M_i in π^* may also be smaller than $c_i^{*'}$. However, we can recursively stretch $p^*_{jo_j}(i_1 \le j \le i_s)$ as little as possible into $p^{*'}_{jo_j}$, such that $\sum_{j=i_1}^{i_j} p^{*'}_{jo_j}$ is an integer power of $1+\varepsilon_0$ and therefore$\sum_{j=i_1}^{i_s} p^{*'}_{jo_j} = c_i^{*'}$. Hence: $c_i \le c_i^{'} \le c_i^{*'}$ $= \sum_{j=i_1}^{i_s} p^{*'}_{jo_j} \le (1+\varepsilon_0)(\sum_{j=i_1}^{i_s-1} p^{*'}_{jo_j} + p^*_{i_s o_i s}) \le (1+\varepsilon_0)(1+\varepsilon_0)(\sum_{j=i_1}^{i_s-2} p^{*'}_{jo_j} + p^*_{i_s-1 o_{i_s-1}} +$ $p^*_{i_s o_{i_s}}) = (1+\varepsilon_0)^2(\sum_{j=i_1}^{i_s-2} p^{*'}_{jo_j} + p^*_{i_s-1 o_{i_s-1}} + p^*_{i_s o_{i_s}}) \le \cdots \le (1+\varepsilon_0)^{(i_s-i_1+1)} \sum_{j=i_1}^{i_s} p^*_{jo_j} \le$ $(1+\varepsilon_0)^n \sum_{j=i_1}^{i_s} p^*_{jo_j} \le (1+\varepsilon) c_i^*$.

We have established the inequality $c_i \le (1+\varepsilon)c_i^*$, which hold for any $1 \le i \le m$, Therefore the makespan of π is at most $(1+\varepsilon)$times that of the optimal one. □

It's not hard to calculate that the running time is $O(((1/\varepsilon)n\log(nP_{\max}))^m nk)$.

Theorem 5. *$P_m|dm|C_{\max}/TPC$ admits an FPTAS.*

5 Conclusion and remarks

In this paper, we design pseudo-polynomial time algorithms by approach of dynamic programming and FPTASs for $1|r_j, dm|C_{\max}/TPC$ and $P_m|dm|C_{\max}/TPC$. For the first problem, our main approach is dynamic programming and scaling-and-rounding; And for the second one, our approach is dynamic programming and geometry partitioning. Can we have the same results on uniform or unrelated parallel machines? And various on-line models are also under investigation.

References

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

[2] Z. Chen, Q. Lu and G. Tang. Single machine scheduling with discretely controllable processing times. *Operations Research Letters*, 21, 69–76, 1997.

[3] B. Chen, C. N. Potts and G. J. Woeginger. A review of machine scheduling: complexity, algorithmthms and approximability. *Handbook of Combinatorial Optimization*, 3, edited by D.Z. Du and P.M. Pardalos, 21–169, Kluwer Academic Publishers, 1998.

[4] R. G. Vickson. Two single machine sequencing problems involving controllable job processing times. *AIIE Transactions*, 12, 258–262, 1980.

[5] R. L. Daniels, J. B. Mazzola. Flow shop scheduling with resource flexibility. *Operations Research*, 42, 504–522, 1994.

[6] Z. Cao, Z. Wang, Y. Zhang and S. Liu. On several scheduling problems with rejection or discretely compressible processing times. *Lecture Notes in Computer Science*, 3959, 90–98, 2006.

[7] C. K. Poon, P. Zhang. Minimizing makespan in batch machine scheduling. *Algorithmica*, 39, 1–20, 2004.