Theoretical Computer Science 411 (2010) 1877–1882

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Single-machine scheduling under the job rejection constraint☆

Liqi Zhang, Lingfa Lu, Jinjiang Yuan *

Department of Mathematics, Zhengzhou University, Zhengzhou, Henan 450001, People's Republic of China

ARTICLE INFO

Article history:
Received 20 July 2009
Received in revised form 1 February 2010
Accepted 7 February 2010
Communicated by D.-Z. Du

Keywords:
Scheduling
Rejection penalty
Fully polynomial-time approximation scheme

ABSTRACT

In this paper, we consider single-machine scheduling problems under the job rejection constraint. A job is either rejected, in which case a rejection penalty has to be paid, or accepted and processed on the single machine. However, the total rejection penalty of the rejected jobs cannot exceed a given upper bound. The objective is to find a schedule such that a given criterion f is minimized, where f is a non-decreasing function on the completion times of the accepted jobs. We analyze the computational complexities of the problems for distinct objective functions and present pseudo-polynomial-time algorithms. In addition, we provide a fully polynomial-time approximation scheme for the makespan problem with release dates. For other objective functions related to due dates, we point out that there is no approximation algorithm with a bounded approximation ratio.

1. Introduction

In most classical scheduling problems, all jobs must be processed on the machines, i.e., rejection is not allowed for the jobs. However, to reduce manufacturing costs and obtain maximum profits, the manufacturer often rejects some jobs which have larger processing times and bring relatively small profits. However, rejecting jobs frequently might lead to low prestige for the manufacturer. Thus, to avoid this, the manufacturer might wish to minimize a given performance criterion under the constraint that the total rejection penalty of the rejected jobs cannot exceed a given upper bound.

Machine scheduling with rejection was first considered by Bartal et al. [1]. They studied a multi-processor scheduling problem to minimize the sum of the makespan of the accepted jobs and the total rejection penalty of the rejected jobs. For the on-line version, they present a best-possible on-line algorithm with a competitive ratio $\frac{\sqrt{5}+3}{2} \approx 2.618$; for the off-line version, they present a polynomial-time approximation scheme. After that, machine scheduling with rejection received more and more attention. For preemption allowed for the accepted jobs, Seiden [23] presented an on-line algorithm with a better competitive ratio $\frac{4+\sqrt{10}}{3} < 2.3874$. Hoogeveen et al. [11] considered the off-line multi-processor scheduling problem with rejection where preemption is allowed. Dosa and He [4] studied the on-line scheduling problem with machine cost and rejection to minimize the sum of the makespan, the cost for purchasing machines, and the total penalty of all rejected jobs. For the small job case, they presented an optimal on-line algorithm with a competitive ratio 2. Engels et al. [6] considered the single-machine scheduling problem with rejection to minimize the sum of the weighted completion times of the accepted jobs and the total penalty of the rejected jobs. They showed that the problem is binary NP-hard and presented a fully polynomial-time approximation scheme. Specifically, they presented a polynomial-time algorithm when all weights of the jobs are identical. Epstein et al. [7] considered the on-line scheduling problem of unit-time jobs with rejection to minimize the sum of the total completion times of the accepted jobs and the total penalty of the rejected jobs. Sengupta [22] considered the single-machine scheduling problem to minimize the sum of the maximum lateness of the

☆ Research supported by NSFC (10971201), NSFC (10901142) and NSFC-RGC (70731160633).
* Corresponding author. Tel.: +86 371 67767835.
E-mail address: yuanjj@zzu.edu.cn (J. Yuan).

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.02.006

accepted jobs and the total penalty of the rejected jobs. Cheng and Sun [3] studied the single-machine scheduling problem with deterioration and rejection, in which the processing time of a job is a linear function of its starting time. Zhang et al. [25] considered the single-machine scheduling problem with release dates and rejection. Lu et al. [19,20] considered the unbounded and bounded parallel-batch machine scheduling problems with release dates and rejection, respectively.

Note that scheduling with rejection is in fact one of the bi-criteria scheduling problems. Bi-criteria (or multi-criteria) scheduling problems have been studied widely in the last two decades. Three surveys of the results in this area were provided by Hoogeveen [10,12] and Lee and Vairakarkis [17]. Generally, given two criteria f and g in a bi-criteria problem, three main directions are studied in the previous literature: (1) minimizing f among the set of optimal schedules for g; (2) minimizing the linear composite objective function $f + g$; (3) minimizing the objective function f given an upper bound on g.

Note that g is fixed as the total rejection penalty of the rejected jobs in our problems. Clearly, minimizing f among the set of optimal schedules for g is equivalent to the classical single-machine scheduling without rejection. Thus, direction (1) is trivial for our problems. To the best of our knowledge, most existing research on scheduling with rejection focuses on direction (2). Hence, it is necessary to study direction (3). Cao et al. [2] considered the single-machine scheduling problem under the job rejection constraint to minimize the total weighted completion times of the accepted jobs. They showed that this problem is binary NP-hard and presented a pseudo-polynomial-time dynamic programming algorithm and a fully polynomial-time approximation scheme. However, other objective functions were not studied in the literature.

2. Problem formulation and preliminaries

The single-machine scheduling problem under the job rejection constraint can be described as follows. There are a single machine and n jobs $J_1, \ldots, J_n$. Each job J_j has a processing time p_j, a weight w_j, a due date d_j and a rejection penalty e_j. We assume that all numbers p_j, w_j, d_j, e_j are non-negative integers. Job J_j is either rejected, in which case a rejection penalty e_j has to be paid, or accepted and processed on the machine. Let A and R be the index set of the accepted jobs and the index set of the rejected jobs, respectively. The objective is to find a schedule such that a given criterion f is minimized under the job rejection constraint $\sum_{j\in R} e_j \le U$, where U is a given upper bound and f is a non-decreasing function of the completion times of the accepted jobs. For example, we can choose f from $\{C_{\max}, L_{\max}, \sum w_jC_j, \sum w_jT_j, \sum w_jU_j\}$. Using the general notation for a scheduling problem, this problem is denoted by $1|\sum_{j\in R} e_j \le U|f$.

If rejection is not allowed, the corresponding problem is denoted by $1||f$. Clearly, problem $1||C_{\max}$ is trivial. For problem $1||L_{\max}$, Lawler [14] showed that the problem can be solved by using the EDD rule (Earliest Due Date first). For problem $1||\sum w_jC_j$, Smith [24] showed that the problem can be solved by using the WSPT rule (Weighted Shortest Processing Time first). If all jobs have identical weight, the WSPT rule is equivalent to the SPT rule (Shortest Processing Time first). For problem $1||\sum U_j$, it can be solved by the famous Moore's algorithm [21]. Problems $1||\sum T_j$ and $1||\sum w_jU_j$ were both proved to be binary NP-hard [5,13] and to be solvable in pseudo-polynomial-time [15,16]. Problem $1||\sum w_jT_j$ was shown by Lenstra et al. [18] to be strongly NP-hard.

In this paper, we consider the single-machine scheduling problem under the job rejection constraint. We analyze the computational complexities of the problems for distinct objective functions and present pseudo-polynomial-time algorithms. In addition, we provide a fully polynomial-time approximation scheme for the makespan problem with release dates. For other objective functions related to due dates, we point out that there is no approximation algorithm with a bounded approximation ratio.

3. NP-hard proofs

Theorem 3.1. *The scheduling problem* $1|\sum_{j\in R} e_j \le U|C_{\max}$ *is equivalent to the minimization knapsack problem. Thus, problem* $1|\sum_{j\in R} e_j \le U|C_{\max}$ *is NP-hard.*

Proof. We use the NP-hard minimization knapsack problem (Güntzer and Jungnickel [9]) for the reduction. Note that $C_{\max} = \sum_{j\in A} p_j$ and $\sum_{j\in A} e_j + \sum_{j\in R} e_j = \sum_{j=1}^{n} e_j$. Thus, minimizing $C_{\max}$ under the constraint $\sum_{j\in R} e_j \le U$ is equivalent to minimizing $\sum_{j\in A} p_j$ under the constraint $\sum_{j\in A} e_j \ge \sum_{j=1}^{n} e_j - U$. And so, problem $1|\sum_{j\in R} e_j \le U|C_{\max}$ is equivalent to the minimization knapsack problem. Thus, problem $1|\sum_{j\in R} e_j \le U|C_{\max}$ is NP-hard, too. □

Corollary 3.2. *Problems* $1|\sum_{j\in R} e_j \le U|L_{\max}$, $1|\sum_{j\in R} e_j \le U|\sum U_j$ *and* $1|\sum_{j\in R} e_j \le U|\sum T_j$ *are NP-hard.*

Proof. For each $j = 1, 2, \ldots, n$, we set $d_j = Y$. Then each one of $L_{\max} \le 0$, $\sum U_j \le 0$ and $\sum T_j \le 0$ is equivalent to $C_{\max} \le Y$. Thus, problems $1|\sum_{j\in R} e_j \le U|L_{\max}$, $1|\sum_{j\in R} e_j \le U|\sum U_j$ and $1|\sum_{j\in R} e_j \le U|\sum T_j$ are NP-hard. □

Remark 3.3. If we set $e_j = U + 1$ for each $j = 1, \ldots, n$, then $1|\sum_{j\in R} e_j \le U|f$ is in fact equivalent to problem $1||f$. Since problem $1||\sum w_jT_j$ is strongly NP-hard, $1|\sum_{j\in R} e_j \le U|\sum w_jT_j$ is strongly NP-hard, too.

Theorem 3.4. *The scheduling problem* $1|\sum_{j\in R} e_j \le U|\sum C_j$ *is NP-hard.*

Proof. The decision version of the problem is clearly in NP. We use the NP-complete even–odd-partition problem (Garey and Johnson [8]) for the reduction.

The even–odd-partition problem: Given $2t+1$ positive integers $a_1, a_2, \ldots, a_{2t}, B$ such that $\sum_{i=1}^{2t} a_i = 2B$, is there a subset $S \subset \{1, 2, \ldots, 2t\}$ such that $|S \bigcap \{2i-1, 2i\}| = 1$ for each $i = 1, 2, \ldots, t$ and $\sum_{i\in S} a_i = B$?

For a given instance of the even–odd-partition problem, we construct an instance of the decision version of problem $1|\sum_{j\in R} e_j \le U|\sum C_j$ as follows.

- $n = 2t$ jobs.
- For each i with $1 \le i \le t$, we define two jobs J_{2i-1} and J_{2i} with

$$p_{2i-1} = \frac{2^{i-1}B + a_{2i-1}}{t-i+1}, \qquad e_{2i-1} = 2^{i-1}B + a_{2i-1}, \qquad p_{2i} = \frac{2^{i-1}B + a_{2i}}{t-i+1}, \qquad e_{2i} = 2^{i-1}B + a_{2i}.$$

- The upper bound is defined by $U = 2^t B$.
- The threshold value is defined by $Y = 2^t B$.
- The decision asks whether there is a schedule π such that $\sum_{j\in A} C_j \le Y$ under the constraint $\sum_{j\in R} e_j \le U$.

It can be observed that the above construction can be done in polynomial time. First, we assume that the even–odd-partition instance has a solution $S \subset \{1, 2, \ldots, 2t\}$ such that $|S \bigcap \{2i-1, 2i\}| = 1$ for each $i = 1, 2, \ldots, t$ and $\sum_{i\in S} a_i = B$. Note that $\max\{p_{2i-1}, p_{2i}\} < \min\{p_{2i+1}, p_{2i+2}\}$ for each $i = 1, 2, \ldots, t-1$. We process the jobs in $\{J_j : j \in S\}$ using the SPT rule (Shortest Processing Time first) and reject all other jobs. It is not hard to check that $\sum_{j\in A} C_j = 2^t B = Y$ and $\sum_{j\in R} e_j = 2^t B = U$.

Now, we suppose that there is a schedule π such that $\sum_{j\in A} C_j \le Y = 2^t B$ and $\sum_{j\in R} e_j \le U = 2^t B$. We are ready to show that the even–odd-partition instance has a solution. We have the following claims.

Claim 1. $|A \bigcap \{2i-1, 2i\}| = |R \bigcap \{2i-1, 2i\}| = 1$ *for each* $i = 1, 2, \ldots, t$.

Otherwise, we can pick the maximum index k such that either $\{2k-1, 2k\} \subseteq A$ or $\{2k-1, 2k\} \subseteq R$. By the definition of k, we have $|A \bigcap \{2i-1, 2i\}| = |R \bigcap \{2i-1, 2i\}| = 1$ for each $i = k+1, k+2, \ldots, t$. Furthermore, by Smith's SPT rule, we can assume that the accepted jobs are processed using the SPT rule in π. Note that $\max\{p_{2i-1}, p_{2i}\} < \min\{p_{2i+1}, p_{2i+2}\}$ for each $i = 1, 2, \ldots, t-1$. If $\{2k-1, 2k\} \subseteq A$, then we have

$$\begin{aligned}
\sum_{j\in A} C_j &\ge (t-k+1)(p_{2k-1} + p_{2k}) + \sum_{i>k, 2i-1\in A} (t-i+1)p_{2i-1} + \sum_{i>k, 2i\in A} (t-i+1)p_{2i} \\
&> 2^{k-1}B + 2^{k-1}B + 2^k B + \cdots + 2^{t-1}B \\
&= 2^t B \\
&= Y,
\end{aligned}$$

a contradiction. If $\{2k-1, 2k\} \subseteq R$, then we have

$$\begin{aligned}
\sum_{j\in R} e_j &\ge e_{2k-1} + e_{2k} + \sum_{i>k, 2i-1\in R} e_{2i-1} + \sum_{i>k, 2i\in R} e_{2i} \\
&> 2^{k-1}B + 2^{k-1}B + 2^k B + \cdots + 2^{t-1}B \\
&= 2^t B \\
&= U,
\end{aligned}$$

a contradiction again. Thus, we have $|A \bigcap \{2i-1, 2i\}| = |R \bigcap \{2i-1, 2i\}| = 1$ for each $i = 1, 2, \ldots, t$.

Claim 2. $\sum_{j\in A} a_j = \sum_{j\in R} a_j = B$.

Since $\sum_{j\in R} e_j = \sum_{i=1}^{t} 2^{i-1}B + \sum_{j\in R} a_j \le U = 2^t B$, we have $\sum_{j\in R} a_j \le B$. Thus, we also have $\sum_{j\in A} a_j \ge B$. Assuming that $\sum_{j\in A} a_j > B$, we have

$$\begin{aligned}
\sum_{j\in A} C_j &= \sum_{2i-1\in A} (t-i+1)p_{2i-1} + \sum_{2i\in A} (t-i+1)p_{2i} \\
&= \sum_{2i-1\in A} (2^{i-1}B + a_{2i-1}) + \sum_{2i\in A} (2^{i-1}B + a_{2i}) \\
&= \sum_{i=1}^{t} 2^{i-1}B + \sum_{j\in A} a_j \\
&> 2^t B \\
&= Y,
\end{aligned}$$

a contradiction. Hence, we have $\sum_{j\in A} a_j = \sum_{j\in R} a_j = B$.

By Claims 1 and 2, A (and also R) is a solution of the even–odd-partition problem. Theorem 3.4 follows.

4. Dynamic programming algorithms

For problem $1|\sum_{j\in R} e_j \le U|\sum w_jC_j$, Cao et al. [2] presented a dynamic programming algorithm. In this section, we present a dynamic programming algorithm for problems $1|r_j, \sum_{j\in R} e_j \le U|C_{\max}$, $1|\sum_{j\in R} e_j \le U|L_{\max}$ and $1|\sum_{j\in R} e_j \le U|\sum w_jU_j$, respectively. Note that $1|r_j, \sum_{j\in R} e_j \le U|C_{\max}$ is an extension of $1|\sum_{j\in R} e_j \le U|C_{\max}$, by introducing release dates of the jobs. Thus, problem $1|r_j, \sum_{j\in R} e_j \le U|C_{\max}$ is also NP-hard. Furthermore, Lenstra et al. [18] showed that problems $1|r_j|L_{\max}$ and $1|r_j|\sum C_j$ are strongly NP-hard. Thus, all problems $1|r_j, \sum_{j\in R} e_j \le U|f$ are strongly NP-hard, where $f \in \{L_{\max}, \sum C_j, \sum T_j, \sum U_j\}$. That is, it is impossible to have any pseudo-polynomial-time algorithm for these problems.

First, we consider the problem $1|r_j, \sum_{j\in R} e_j \le U|C_{\max}$. Sort the jobs such that $r_1 \le \cdots \le r_n$. Let $f_j(t)$ be the minimum value of the total rejection penalty when the jobs under consideration are $J_1, \ldots, J_j$ and the makespan of the accepted jobs among $J_1, \ldots, J_j$ is exactly t. Now, we consider any optimal schedule for the jobs $J_1, \ldots, J_j$ in which the makespan of the accepted jobs among $J_1, \ldots, J_j$ is exactly t. In any such schedule, there are two possible cases: either job J_j is rejected or job J_j is accepted and processed on the machine.

Case 1. Job J_j is rejected. In this case, the makespan of the accepted jobs among $J_1, \ldots, J_{j-1}$ is still t. Thus, we have $f_j(t) = f_{j-1}(t) + e_j$.

Case 2. Job J_j is accepted. In this case, we have $t \ge r_j + p_j$. If $t > r_j + p_j$, then the makespan of the accepted jobs among $J_1, \ldots, J_{j-1}$ is exactly $t - p_j$. Thus, we have $f_j(t) = f_{j-1}(t - p_j)$. If $t = r_j + p_j$, then the makespan of the accepted jobs among $J_1, \ldots, J_{j-1}$ is at most r_j. Thus, we have $f_j(t) = \min\{f_{j-1}(t') : 0 \le t' \le r_j\}$.

Combining the above two cases, we have the following dynamic programming algorithm DP1.

Dynamic programming algorithm DP1

The boundary conditions:

$$f_1(t) = \begin{cases} e_1, & \text{if } t = 0; \\ 0, & \text{if } t = r_1 + p_1; \\ +\infty, & \text{otherwise.} \end{cases}$$

The recursive function:

$$f_j(t) = \begin{cases} f_{j-1}(t) + e_j, & \text{if } t < r_j + p_j; \\ \min\{f_{j-1}(t) + e_j, \min\{f_{j-1}(t') : 0 \le t' \le r_j\}\}, & \text{if } t = r_j + p_j; \\ \min\{f_{j-1}(t) + e_j, f_{j-1}(t - p_j)\}, & \text{if } t > r_j + p_j. \end{cases}$$

The optimal value is given by $\min\{t : 0 \le t \le r_n + \sum_{j=1}^{n} p_j \text{ and } f_n(t) \le U\}$.

Theorem 4.1. *Algorithm DP1 solves* $1|r_j, \sum_{j\in R} e_j \le U|C_{\max}$ *in* $O(n(r_n + \sum_{j=1}^{n} p_j))$ *time.*

Remark 4.2. If all jobs have the same processing time, i.e., $p_j = p$ for each $j = 1, \ldots, n$, we have $t = 0$ or $t \in \{r_j + kp : 1 \le j, k \le n\}$. Thus, if all jobs have the same processing time, the corresponding problem can be solved by algorithm DP1 in $O(n^3)$ time. It is not hard to present a dual dynamic programming algorithm, similar to algorithm DP1, that solves this problem in $O(n\sum_{j=1}^{n} e_j)$ time. If all jobs have the same rejection penalty, then the corresponding problem can be solved in $O(n^2)$ time.

Now, we consider problem $1|\sum_{j\in R} e_j \le U|L_{\max}$. Sort the jobs such that $d_1 \le \cdots \le d_n$. Let $f_j(t, E)$ be the optimal value of the objective function under the following constraints: (1) the jobs under consideration are $J_1, \ldots, J_j$; (2) the makespan of the accepted jobs among $J_1, \ldots, J_j$ is exactly t; (3) the value of the total rejection penalty among $J_1, \ldots, J_j$ is exactly E.

Dynamic programming algorithm DP2

The boundary conditions:

$$f_1(t, E) = \begin{cases} p_1 - d_1, & \text{if } t = p_1 \text{ and } E = 0; \\ 0, & \text{if } t = 0 \text{ and } E = e_1; \\ +\infty, & \text{otherwise.} \end{cases}$$

The recursive function:

$$f_j(t) = \begin{cases} f_{j-1}(t, E - e_j), & \text{if } f_{j-1}(t - p_j, E) = +\infty; \\ \min\{f_{j-1}(t, E - e_j), \max\{f_{j-1}(t - p_j, E), t - d_j\}\}, & \text{if } f_{j-1}(t - p_j, E) < +\infty. \end{cases}$$

The optimal value is given by $\min\{f_n(t, E) : 0 \le t \le \sum_{j=1}^{n} p_j \text{ and } E \le U\}$.

Theorem 4.3. *Algorithm DP2 solves* $1|\sum_{j\in R} e_j \le U|L_{\max}$ *in* $O(nU\sum_{j=1}^{n} p_j)$ *time.*

Next, we present a dynamic programming algorithm DP3 for problem $1|\sum_{j\in R} e_j \le U|\sum w_jU_j$. Clearly, for problem $1|\sum_{j\in R} e_j \le U|\sum w_jU_j$, there is an optimal schedule such that the early jobs among the accepted jobs are processed first in EDD order, followed by the tardy jobs. Let $f_j(t, E)$ be the optimal value of the objective function under the following constraints: (1) the jobs under consideration are $J_1, \ldots, J_j$; (2) the makespan of the early jobs in the accepted jobs is exactly t; (3) the value of the total rejection penalty is E.

Dynamic programming algorithm DP3

The boundary conditions:

$$f_1(t, E) = \begin{cases} 0, & \text{if } t = 0 \text{ and } E = e_1; \\ 0, & \text{if } t = p_1 \le d_1 \text{ and } E = 0; \\ w_1, & \text{if } t = 0 \text{ and } E = 0; \\ +\infty, & \text{otherwise.} \end{cases}$$

The recursive function:

$$f_j(t, E) = \min\{f_{j-1}(t, E - e_j), f_{j-1}(t - p_j, E), f_{j-1}(t, E) + w_j\}.$$

The optimal value is given by $\min\{f_n(t, E) : 0 \le t \le \sum_{j=1}^{n} p_j \text{ and } E \le U\}$.

Theorem 4.4. *Algorithm DP3 solves* $1|\sum_{j\in R} e_j \le U|\sum w_jU_j$ *in* $O(nU\sum p_j)$ *time.*

5. Fully polynomial-time approximation schemes

By Corollary 3.2, decision problems $1|\sum_{j\in R} e_j \le U|L_{\max} \le 0$, $1|\sum_{j\in R} e_j \le U|\sum T_j \le 0$ and $1|\sum_{j\in R} e_j \le U|\sum U_j \le 0$ are NP-complete. Thus, unless $P = NP$, there is no approximation algorithm for these problems with a bounded approximation ratio. For problem $1|\sum_{j\in R} e_j \le U|\sum w_jC_j$, Cao et al. [2] presented a fully polynomial-time approximation scheme.

In this section, for problem $1|r_j, \sum_{j\in R} e_j \le U|C_{\max}$, we present a fully polynomial-time approximation scheme. By Theorem 3.1, problem $1|\sum_{j\in R} e_j \le U|C_{\max}$ is equivalent to the minimization knapsack problem. For the latter problem, Güntzer and Jungnickel [9] presented a simple 2-approximation algorithm and a fully polynomial-time approximation scheme. However, it is not hard to verify, on the basis of the approximation algorithm for the minimization knapsack problem, that the corresponding approximation algorithm for $1|\sum_{j\in R} e_j \le U|C_{\max}$ cannot guarantee the same approximation ratio. Thus, to obtain a fully polynomial-time approximation scheme for the harder problem $1|r_j, \sum_{j\in R} e_j \le U|C_{\max}$, we have to adopt a new strategy.

If $r_j = p_j = 0$, then there exists an optimal schedule such that J_j is accepted and processed at time 0. Thus, without loss of generality, we assume that $r_j + p_j > 0$ for all $j = 1, \ldots, n$. Assume that π^* is an optimal schedule of a given instance I for problem $1|r_j, \sum_{j\in R} e_j \le U|C_{\max}$. Let A^* and R^* be the index set of the accepted jobs and the index set of the rejected jobs in π^*, respectively. Write $\Delta^* = \max\{r_j + p_j : j \in A^*\}$. For any job J_j with $r_j + p_j > \Delta^*$, we have $j \in R^*$ and $\sum_{r_j+p_j>\Delta^*} e_j \le \sum_{j\in R} e_j \le U$. Let $I' = \{J_j \in I : r_j + p_j \le \Delta^*\}$ and $U' = U - \sum_{r_j+p_j>\Delta^*} e_j$. Then the original problem for instance I is equivalent to the makespan problem under the constraint $\sum_{j\in R} e_j \le U'$ for instance I'.

Given any ϵ with $\epsilon > 0$. Set $\delta = \frac{\epsilon\Delta^*}{n+1}$. For each job $J_j \in I'$, we modify its release date and processing time such that $r_j'' = \lceil \frac{r_j}{\delta} \rceil \delta$ and $p_j'' = \lceil \frac{p_j}{\delta} \rceil \delta$. By the modification, we obtain a new instance I'' such that $r_j \le r_j'' < r_j + \delta$ and $p_j \le p_j'' < p_j + \delta$. For any instance I, let $C^*(I)$ be the optimal makespan value. Furthermore, we have the following lemma.

Lemma 5.1. $C^*(I'') \le (1 + \epsilon)C^*(I)$.

Proof. Assume that π^* is an optimal schedule for instance I (and also for I'). For each accepted job J_j in π^*, we replace p_j by p_j'' and delay its processing by a time length δ. This yields a feasible schedule for the instance I'' with the makespan at most $C^*(I) + (n + 1)\delta \le C^*(I) + \epsilon\Delta^* \le (1 + \epsilon)C^*(I)$. Thus, we have $C^*(I'') \le (1 + \epsilon)C^*(I)$. Lemma 5.1 follows. □

Since the makespan in π^* is at most $\sum_{j\in A^*}(r_j + p_j) \le n\Delta^*$, by Lemma 5.1, the optimal makespan for instance I'' is at most $(1 + \epsilon)n\Delta^*$. Note that all release dates and processing times in I'' are multiples of δ. Thus, we can assume that each accepted job J_j in I'' must complete its processing at some time point $k\delta$, where $1 \le k \le \frac{(1+\epsilon)n\Delta^*}{\delta} = \frac{n(n+1)(1+\epsilon)}{\epsilon}$. Using algorithm DP1 for instance I'', we can obtain an optimal schedule for instance I'' in $O(\frac{n^3}{\epsilon})$ time. Replacing all p_j'' by p_j, we can obtain a feasible schedule for instance I with the makespan at most $C^*(I'') \le (1 + \epsilon)C^*(I)$.

From the above discussion, once Δ^* is determined, we can obtain in $O(\frac{n^3}{\epsilon})$ time a $(1 + \epsilon)$-approximation schedule for instance I. Note that Δ^* has at most n distinct choices. Thus, we can enumerate all possibilities and select the feasible schedule with the minimum makespan value. On the basis of this idea, we present a fully polynomial-time approximation scheme for our problem.

Approximation scheme A_ϵ

Step 1: For each $k = 1, \ldots, n$, set $\Delta^* = r_k + p_k$. Reject all jobs J_j with $r_j + p_j > \Delta^*$. Set $I' = \{J_j : r_j + p_j \leq \Delta^*\}$ and $U' = U - \sum_{r_j+p_j>\Delta^*} e_j$.

Step 2: Set $\delta = \frac{\epsilon\Delta^*}{n+1}$. For each job $J_j \in I'$, we modify its release date and processing time such that $r_j'' = \lceil \frac{r_j}{\delta} \rceil \delta$ and $p_j'' = \lceil \frac{p_j}{\delta} \rceil \delta$. We denote the new instance by I''.

Step 3: If $U' \geq 0$, then apply algorithm DP1 to the instance I'' under the constraint $\sum_{j\in R} e_j \leq U'$. For the schedule obtained from DP1, we replace all p_j'' by p_j to obtain a feasible schedule for instance I.

Step 4: For each $\Delta^* = r_k + p_k$, let C^k be the makespan obtained from Step 3. Select the corresponding schedule with the minimum makespan value.

Let $C^{A_\epsilon}(I)$ be the makespan value obtained from algorithm A_ϵ for any instance I. From the above discussion, we have the following theorem.

Theorem 5.2. *For any instance I, we have $C^{A_\epsilon}(I) \leq (1+\epsilon)C^*(I)$ and the time complexity of algorithm A_ϵ is $O(\frac{n^4}{\epsilon})$. That is, algorithm A_ϵ is a fully polynomial-time approximation scheme for problem $1|r_j, \sum_{j\in R} e_j \leq U|C_{\max}$.*

References

[1] Y. Bartal, S. Leonardi, A.M. Spaccamela, J. Sgall, L. Stougie, Multiprocessor scheduling with rejection, SIAM Journal on Discrete Mathematics 13 (2000) 64–78.
[2] Z.G. Cao, Z. Wang, Y.Z. Zhang, S.P. Liu, On several scheduling problems with rejection or discretely compressible processing times, in: Lecture Notes in Computer Science, vol. 3959, 2006, pp. 90–98.
[3] Y.S. Cheng, S.J. Sun, Scheduling linear deteriorating jobs with rejection on a single machine, European Journal of Operational Research 194 (2009) 18–27.
[4] G. Dosa, Y. He, Scheduling with machine cost and rejection, Journal of Combinatorial Optimization 12 (2006) 337–350.
[5] J.Z. Du, J.Y.-T. Leung, Minimizing total tardiness on one machine is NP-hard, Mathematics of Operations Research 15 (1990) 483–495.
[6] D.W. Engels, D.R. Karger, S.G. Kolliopoulos, S. Sengupta, R.N. Uma, J. Wein, Techniques for scheduling with rejection, Journal of Algorithms 49 (2003) 175–191.
[7] L. Epstein, J. Noga, G.J. Woeginger, On-line scheduling of unit time jobs with rejection: minimizing the total completion time, Operations Research Letters 30 (2002) 415–420.
[8] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, CA, 1979.
[9] M.M. Güntzer, D. Jungnickel, Approximate minimization algorithms for the 0/1 knapsack and subset-sum problem, Operations Research Letters 26 (2000) 55–66.
[10] J.A. Hoogeveen, Single-machine bicriteria scheduling, Ph.D thesis, CWI, Amsterdam, 1992.
[11] H. Hoogeveen, M. Skutella, G.J. Woeginger, Preemptive scheduling with rejection, Mathematics Programming 94 (2003) 361–374.
[12] H. Hoogeveen, Multicriteria scheduling, European Journal of Operational Research 167 (2005) 592–613.
[13] R.M. Karp, Reducibility among combinatorial problems, in: Complexity of Computer Computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972) New York, 1972, pp. 85–103.
[14] E.L. Lawler, Optimal Sequencing of a single machine subject to precedence constraints, Management Science 19 (1973) 544–546.
[15] E.L. Lawler, A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness, Annals of Discrete Mathematics 1 (1977) 331–342.
[16] E.L. Lawler, J.M. Moore, A functional equation and its application to resource allocation and sequencing problems, Management Science 16 (1978) 77–84.
[17] C.-Y. Lee, G. Vairaktarakis, Single machine dual criteria scheduling: A survey, in: P.M. Pardalos (Ed.), Complexity in Numerical Optimization, World Scientific, River Edge, NJ, 1993, pp. 269–298.
[18] J.K. Lenstra, A.H.G. Rinnooy Kan, P. Brucker, Complexity of machine scheduling problems, Annals of Discrete Mathematics 1 (1977) 343–362.
[19] L.F. Lu, L.Q. Zhang, J.J. Yuan, The unbounded parallel batch machine scheduling with release dates and rejection to minimize makespan, Theoretical Computer Science 396 (2008) 283–289.
[20] L.F. Lu, T.C.E. Cheng, J.J. Yuan, L.Q. Zhang, Bounded single-machine parallel-batch scheduling with release dates and rejection, Computers and Operation Research 36 (2009) 2748–2751.
[21] J.M. Moore, An n job, one machine sequencing algorithm for minimizing the number of late jobs, Management Science 15 (1968) 102–109.
[22] S. Sengupta, Algorithms and approximation schemes for minimum lateness/tardiness scheduling with rejection, Lecture Notes in Computer Science 2748 (2003) 79–90.
[23] S. Seiden, Preemptive multiprocessor scheduling with rejection, Theoretical Computer Science 262 (2001) 437–458.
[24] W.E. Smith, Various optimizers for single-stage production, Naval Research Logistics Quarterly 1 (1956) 59–66.
[25] L.Q. Zhang, L.F. Lu, J.J. Yuan, Single machine scheduling with release dates and rejection, European Journal of Operational Research 198 (2009) 975–978.