Available online at www.sciencedirect.com
ScienceDirect

Theoretical Computer Science 396 (2008) 283–289

Theoretical Computer Science
www.elsevier.com/locate/tcs

Note

The unbounded parallel batch machine scheduling with release dates and rejection to minimize makespan[☆]

Lingfa Lu, Liqi Zhang, Jinjiang Yuan[*]

Department of Mathematics, Zhengzhou University, Zhengzhou, Henan 450052, People's Republic of China

Received 21 November 2007; received in revised form 11 February 2008; accepted 12 February 2008

Communicated by D.-Z. Du

Abstract

In this paper, we consider the unbounded parallel batch machine scheduling with release dates and rejection. A job is either rejected with a certain penalty having to be paid, or accepted and processed in batches on the parallel batch machine. The processing time of a batch is defined as the longest processing time of the jobs contained in it. The objective is to minimize the sum of the makespan of the accepted jobs and the total rejection penalty of the rejected jobs. We show that this problem is binary NP-hard and provide a pseudo-polynomial-time algorithm. When the jobs have the same rejection penalty, the problem can be solved in polynomial time. Finally, a 2-approximation algorithm and a fully polynomial-time approximation scheme are given for the problem.
© 2008 Published by Elsevier B.V.

Keywords: Scheduling; Rejection penalty; Fully polynomial-time approximation scheme

1. Introduction

The unbounded parallel batch machine scheduling problem with release dates and rejection can be described as follows. There are n jobs $J_1, \dots, J_n$ with each job J_j having a processing time p_j, a release date r_j, and a rejection penalty w_j. Each job J_j is either rejected with a rejection penalty w_j having to be paid, or accepted and processed on the parallel batch machine. The parallel batch machine can process a number of jobs simultaneously as a batch. The jobs in the same batch have the same starting time and completion time. The processing time of a batch is defined as the longest processing time of the jobs contained in it. The objective is to minimize the sum of makespan of the accepted jobs and total rejection penalty of the rejected jobs. Let R be the set of the rejected jobs. Using the general notation for scheduling problem introduced by Graham et al. [9], the problem is denoted by $1|\text{p-batch}, r_j|C_{\max} + \sum_{J_j \in R} w_j$.

In the last decade, there has been significant interest in scheduling problems that involve an element of batching. The motivation for batching jobs is a gain in efficiency: it may be cheaper and faster to process jobs in a batch than to

[☆] Research supported by NSFC (10671183), NSFC-RGC (70731160633) and SRFDP (20070459002).
[*] Corresponding author. Tel.: +86 371 67767835.
E-mail address: yuanjj@zzu.edu.cn (J. Yuan).

process them individually. The fundamental model of parallel batch machine scheduling was first introduced by Lee et al. [11] with the restriction that the number of the jobs in each batch is bounded by a number b. This bounded model is motivated by the burn-in operations in semiconductor manufacturing. For example, a batch of integrated circuits (jobs) may be put inside an oven of limited size to test for their thermal standing ability. The circuits are heated inside the oven until all circuits are burned. The burn-in time of the circuits (job processing times) may be different. When a circuit is burned, it has to wait inside the oven until all circuits are burned. Therefore, the processing time of a batch of circuits is the longest processing time of the jobs in the batch. Brucker et al. [1] provided an extensive discussion of the unbounded version of the parallel batch machine scheduling. For the problem $1|\text{p-batch}, r_j|C_{\max}$, Lee and Uzsoy [12] proposed a polynomial-time dynamic programming algorithm. Deng and Zhang [5] proved that the problem $1|p\text{-batch}, r_j|\sum w_j C_j$ is binary NP-hard. Cheng et al. [4] proved that the problem $1|\text{p-batch}, r_j|L_{\max}$ is binary NP-hard. Liu et al. [13] provided a pseudo-polynomial-time algorithm for the problem $1|\text{p-batch}, r_j|f$, where $f \in \{C_{\max}, L_{\max}, \sum(w_j)C_j, \sum(w_j)U_j, \sum(w_j)T_j\}$. In addition, more recent developments on this topic can be found in [2].

In classical scheduling literatures, all jobs must be processed and no rejection is allowed. However, in real applications, this may not be true. Due to the limited resources, the scheduler can have the option to reject some jobs. The machine scheduling problem with rejection was first introduced by Bartal et al. [3]. They studied the off-line version as well as the on-line version on identical parallel machines. The objective is to minimize the sum of makespan of the accepted jobs and total rejection penalty of the rejected jobs. After that, the machine scheduling problem with rejection received more and more attentions. Seiden [14] presented a better on-line algorithm if preemption is allowed to all jobs. Hoogeveen, Skutella and Woeginger [10] considered the off-line multi-processor scheduling problem with rejection when preemption is allowed. Zhang et al. [15] proved the single machine scheduling problem with release date and rejection to minimize makespan is binary NP-hard. They also proposed a pseudo-polynomial-time algorithm and a fully polynomial-time approximation scheme for the problem. Engels et al. [6] considered the single machine scheduling with rejection to minimize the sum of weighted completion times of the scheduled jobs and total rejection penalty of the rejected jobs. Epstein, Noga and Woeginger [7] considered on-line scheduling problem of unit-time jobs with rejection to minimize the total completion time.

In this paper, we consider the unbounded parallel batch machine scheduling with release dates and rejection. The objective is to minimize the sum of makespan of the accepted jobs and total rejection penalty of the rejected jobs. We show that this problem is binary NP-hard and provide a pseudo-polynomial-time algorithm. When the jobs have the same rejection penalty, the problem can be solved in polynomial time. Finally, a 2-approximation algorithm and a fully polynomial-time approximation scheme are given for the problem.

2. NP-hardness proof

Theorem 2.1. *The scheduling problem* $1|p\text{-batch}, r_j|C_{\max} + \sum_{J_j \in R} w_j$ *is binary NP-hard.*

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

Partition problem: Given t positive integers $a_1, a_2, \ldots, a_t$ with $\sum_{i=1}^{t} a_i = 2B$, is there a subset $S \subset \{a_1, a_2, \ldots, a_t\}$ such that $\sum_{a_i \in S} a_i = B$?

For a given instance of the Partition problem, we construct an instance of the decision version of problem $1|\text{p-batch}, r_j|C_{\max} + \sum_{J_j \in R} w_j$ as follows.

- $n = 2t + 1$ jobs.
- For $i = 1$, we have a "heavy" job J_1 with $(r_1, p_1, w_1) = (0, 2^{t+1}B, 2^{t+2}B)$ and a "light" job J_2 with $(r_2, p_2, w_2) = (0, 2^{t+1}B + 2a_1, a_1)$.
- For each $2 \le i \le t$, we have a "heavy" job J_{2i-1} with

$$(r_{2i-1}, p_{2i-1}, w_{2i-1}) = \left(\sum_{k=1}^{i-1} 2^{t+2-k}B, 2^{t+2-i}B, 2^{t+2}B \right)$$

and a "light" job J_{2i} with $(r_{2i}, p_{2i}, w_{2i}) = (\sum_{k=1}^{i-1} 2^{t+2-k}B, 2^{t+2-i}B + 2a_i, a_i)$.

- For $j = 2t + 1$, we only have a "heavy" job J_{2t+1} with

$$(r_{2t+1}, p_{2t+1}, w_{2t+1}) = ((2^{t+2} - 2)B, 0, 2^{t+2}B).$$

- The threshold value is defined by $Y = (2^{t+2} - 1)B$.
- The decision asks whether there is a schedule π such that $C_{\max} + \sum_{J_j \in R} w_j \le Y$.

It can be observed that the above construction can be done in polynomial time. Assume first that the Partition problem has a solution S such that $\sum_{a_i \in S} a_i = B$. We construct a schedule such that $C_{\max} + \sum_{J_j \in R} w_j \le Y$ by the following way: If $a_i \in S$, we assign the jobs J_{2i-1} and J_{2i} as a batch B_i. If $a_i \notin S$, we assign the job J_{2i-1} as a batch B_i. We also assign the job J_{2t+1} as a batch B_{t+1}. Reject all other jobs and process the batches in the order of $B_1, \ldots, B_{t+1}$. It is easy to verify that

$$C_{\max} + \sum_{J_j \in R} w_j = \sum_{a_i \in S} (2^{t+2-i}B + 2a_i) + \sum_{a_i \notin S} 2^{t+2-i}B + \sum_{a_i \notin S} a_i = (2^{t+2} - 1)B = Y.$$

Conversely, suppose that there is a schedule π such that $C_{\max} + \sum_{J_j \in R} w_j \le Y$. We are ready to show that the Partition problem has a solution. Denote by A and R the sets of accepted jobs and rejected jobs, respectively. We have the following claims.

Claim 1. *$J_{2i-1} \in A$ for each $1 \le i \le t + 1$ and $C_{\max} \ge (2^{t+2} - 2)B$.*

Proof. If there exists some job $J_{2i-1} \in R$ with $1 \le i \le t+1$, then we have $C_{\max} + \sum_{J_j \in R} w_j \ge w_{2i-1} = 2^{t+2}B > Y$, a contradiction. Thus, we have $J_{2i-1} \in A$ for each $1 \le i \le t + 1$. Since $J_{2t+1} \in A$, we have $C_{\max} \ge r_{2t+1} = (2^{t+2} - 2)B$. □

Claim 2. *For each pair i and j with $1 \le i < j \le t + 1$, J_{2i-1} and J_{2j-1} cannot be contained in the same batch.*

Proof. Otherwise, there exist two jobs J_{2i-1} and J_{2j-1} $(i < j)$ contained in the same batch. Then we have

$$C_{\max} \ge r_{2j-1} + p_{2i-1} \ge r_{2i+1} + p_{2i-1} \ge \sum_{k=1}^{i} 2^{t+2-k}B + 2^{t+2-i}B = 2^{t+2}B > Y,$$

a contradiction. □

By Claim 2, all "heavy" jobs $\{J_{2i-1} : 1 \le i \le t+1\}$ must belong to $t+1$ distinct batches. For $1 \le i \le t+1$, let B_i be the batch containing J_{2i-1} in π. Furthermore, by Claim 1, only the "light" jobs $\{J_{2i} : 1 \le i \le t\}$ can be rejected.

Claim 3. *If $J_{2i} \in A$, then we have $J_{2i} \in B_i$.*

Proof. Assume to the contrary that there exists a job $J_{2i} \in A$ such that $J_{2i} \notin B_i$. If $J_{2i} \notin B_k$ for all $1 \le k \le t + 1$, then we have

$$C_{\max} \ge \sum_{k=1}^{t} p_{2k-1} + p_{2i} \ge \sum_{k=1}^{t} 2^{t+2-k}B + 2^{t+2-i}B \ge \sum_{k=1}^{t} 2^{t+2-k}B + 4B = 2^{t+2}B > Y,$$

a contradiction. If $J_{2i} \in B_k$ for some k with $1 \le k \le t + 1$ and $k \ne i$, we consider two possibilities. When $i < k$, we have

$$C_{\max} \ge r_{2k-1} + p_{2i} \ge r_{2i+1} + p_{2i} \ge \sum_{j=1}^{i} 2^{t+2-j}B + 2^{t+2-i}B = 2^{t+2}B > Y,$$

a contradiction. When $i > k$, we have

$$C_{\max} \ge r_{2i} + p_{2k-1} \ge r_{2k+2} + p_{2k-1} = \sum_{j=1}^{k} 2^{t+2-j}B + 2^{t+2-k}B = 2^{t+2}B > Y,$$

a contradiction again. □

By Claims 2 and 3, we have $B_i = \{J_{2i-1}, J_{2i}\}$ if $J_{2i} \in A$, and $B_i = \{J_{2i-1}\}$ if $J_{2i} \in R$. Let $S = \{a_i : J_{2i} \in R\}$. We are ready to show that S is a solution of the Partition problem.

Since $C_{\max} \geq (2^{t+2} - 2)B$ and $C_{\max} + \sum_{J_j \in R} w_j \leq Y = (2^{t+2} - 1)B$, we have $\sum_{a_i \in S} a_i = \sum_{J_{2i} \in R} w_{2i} \leq B$. If $\sum_{a_i \in S} a_i < B$, then we have

$$\begin{aligned} C_{\max} + \sum_{J_j \in R} w_j &\geq \sum_{J_{2i} \in A} p_{2i} + \sum_{J_{2i} \in R} p_{2i-1} + \sum_{J_{2i} \in R} w_{2i} \\ &= \sum_{a_i \notin S} (2^{t+2-i}B + 2a_i) + \sum_{a_i \in S} 2^{t+2-i}B + \sum_{a_i \in S} a_i \\ &= \sum_{i=1}^{t} 2^{t+2-i}B + 2\left(\sum_{a_i \notin S} a_j + \sum_{a_i \in S} a_j\right) - \sum_{a_i \in S} a_j \\ &> \sum_{i=1}^{t} 2^{t+2-i}B + 2B + B \\ &= (2^{t+2} - 1)B \\ &= Y, \end{aligned}$$

a contradiction. Hence, we have $\sum_{a_i \in S} a_i = B$, and so S is a solution of partition problem. Theorem 2.1 follows. □

3. A dynamic programming algorithm

For a schedule π, we say that the accepted jobs are processed in the LPT-rule in π if, for every two accepted jobs J_i and J_j, $p_i > p_j$ implies that J_i is processed no later than J_j in π.

Lemma 3.1. *For problem* $1|p\text{-}batch, r_j|C_{\max} + \sum_{J_j \in R} w_j$, *there exists an optimal schedule such that the accepted jobs are processed in the LPT-rule.*

Proof. Otherwise, there exist two accepted jobs J_i and J_j with $p_i > p_j$ such that J_i is processed later than J_j. Then we put J_i into the batch containing J_j. Clearly, this does increasing the objective value. A finite number of repetitions of this procedure yields an optimal schedule of the required form. □

Based on Lemma 3.1, we only consider the schedules in which the accepted jobs are processed in the LPT-rule. Assume that jobs have been indexed such that $p_1 \geq \cdots \geq p_n$. We first introduce two useful notations.

- $A_j(k, W)$ is the optimal objective function value of the following scheduling problem: (1) The jobs in consideration are $J_1, \ldots, J_j$. (2) J_j is accepted. (3) In the last batch, J_k is the job with the minimum index. (4) the total rejection penalty is exactly W.
- $R_j(k, W)$ is the optimal objective function value of the following scheduling problem: (1) The jobs in consideration are $J_1, \ldots, J_j$. (2) J_j is rejected. (3) In the last batch, J_k is the job with the minimum index. (4) the total rejection penalty is exactly W.

In an optimal schedule for the first j jobs which assumes either $A_j(k, W)$ or $R_j(k, W)$, we distinguish the following four possibilities for J_{j-1} and J_j.

Case 1. Both J_{j-1} and J_j are rejected.

Since J_j is rejected, in the corresponding optimal schedule for $J_1, \ldots, J_{j-1}$, J_k is still the job with the minimum index in the last batch and the total rejection penalty among $J_1, \ldots, J_{j-1}$ is $W - w_j$. Thus, we have $R_j(k, W) = R_{j-1}(k, W - w_j) + w_j$ since J_{j-1} is rejected.

Case 2. J_{j-1} is accepted and J_j is rejected.

Similar to case 1, we have $R_j(k, W) = A_{j-1}(k, W - w_j) + w_j$.

Case 3. J_{j-1} is rejected and J_j is accepted.

If $k = j$, then J_j consists the last batch. Thus, we have $A_j(j, W) = \min_{0 \le k' \le j-2}\{\max\{R_{j-1}(k', W) - W, r_j\} + p_j + W\}$. If $k < j$, then both J_k and J_j belong to the last batch. Thus, we have $A_j(k, W) = \max\{R_{j-1}(k, W) - W - p_k, r_j\} + p_k + W$. In conclusion, we have

$$A_j(k, W) = \begin{cases} \min\limits_{0 \le k' \le j-2}\{\max\{R_{j-1}(k', W) - W, r_j\} + p_j + W\}, & \text{if } k = j; \\ \max\{R_{j-1}(k, W) - W - p_k, r_j\} + p_k + W, & \text{otherwise.} \end{cases}$$

Case 4. Both J_{j-1} and J_j are accepted.

Similar to case 3, we have

$$A_j(k, W) = \begin{cases} \min\limits_{1 \le k' \le j-1}\{\max\{A_{j-1}(k', W) - W, r_j\} + p_j + W\}, & \text{if } k = j; \\ \max\{A_{j-1}(k, W) - W - p_k, r_j\} + p_k + W, & \text{otherwise.} \end{cases}$$

Combining the above four cases, we have the following dynamic programming algorithm DPA.

Dynamic programming algorithm DPA

The boundary conditions:

$$A_1(1, 0) = r_1 + p_1 \quad \text{and} \quad A_1(k, W) = \infty \quad \text{for any } k \ne 1 \text{ or } W \ne 0.$$
$$R_1(0, w_1) = w_1 \quad \text{and} \quad R_1(k, W) = \infty \quad \text{for any } k \ne 0 \text{ or } W \ne w_1.$$

The recursive function: If $k = j$, then

$$A_j(j, W) = \min \begin{cases} \min\limits_{0 \le k' \le j-2}\{\max\{R_{j-1}(k', W) - W, r_j\} + p_j + W\}, \\ \min\limits_{1 \le k' \le j-1}\{\max\{A_{j-1}(k', W) - W, r_j\} + p_j + W\}. \end{cases}$$

If $k < j$, then

$$A_j(k, W) = \min \begin{cases} \max\{R_{j-1}(k, W) - W - p_k, r_j\} + p_k + W, \\ \max\{A_{j-1}(k, W) - W - p_k, r_j\} + p_k + W. \end{cases}$$

Furthermore,

$$R_j(k, W) = \min\{A_{j-1}(k, W - w_j) + w_j, R_{j-1}(k, W - w_j) + w_j\}.$$

The optimal value is given by

$$\min\left\{\min\{A_n(k, W), R_n(k, W)\} : 0 \le k \le n \text{ and } 0 \le W \le \sum w_j\right\}.$$

Theorem 3.2. *The dynamic programming algorithm DPA solves* $1|p\text{-batch}, r_j|C_{\max} + \sum_{J_j \in R} w_j$ *in* $O(n^2 \sum w_j)$ *time.*

Proof. The correctness of the algorithm is guaranteed by the above discussion. The recursive function has at most $O(n^2 \sum w_j)$ states. If $k = j$, each iteration costs $O(n)$ time; otherwise, each iteration costs a constant time. Hence, the total running time is bounded by $O(n^2 \sum w_j)$. $\square$

Specifically, if $w_j = w$ for $j = 1, \ldots, n$, then $W \in \{jw : 1 \le j \le n\}$. That is, there are at most n choices for each W in the above DPA. Thus, we have the following corollary.

Corollary 3.3. *The dynamic programming algorithm DPA solves* $1|p\text{-batch}, r_j, w_j = w|C_{\max} + \sum_{J_j \in R} w_j$ *in* $O(n^3)$ *time.*

4. Approximation algorithms

4.1. A 2-approximation algorithm

Assume that S is a set of jobs. We use $p(S) = \max_{J_j \in S} p_j$ and $w(S) = \sum_{J_j \in S} w_j$ to denote the processing time and the total rejection penalty of S, respectively. Now, we propose a 2-approximation algorithm for the considered problem.

Approximation algorithm A

Step 1. For each $t \in \{r_j : j = 1, \dots, n\}$ and $p \in \{p_j : j = 1, \dots, n\}$, we divide the jobs into two sets of jobs such that $S_1(t, p) = \{J_j : r_j \le t$ and $p_j \le p\}$ and $S_2(t, p) = \{J_j : r_j > t$ or $p_j > p\}$.

Step 2. For each $S_1(t, p)$ and $S_2(t, p)$ obtained from Step 1, we accept and process all jobs in $S_1(t, p)$ as a batch starting at time t on the machine, and reject the jobs in $S_2(t, p)$. The resulting schedule is denoted by $\pi(t, p)$.

Step 3. Let $Z(t, p)$ be the value of the objective function for each $\pi(t, p)$. Among all the schedules obtained above, select the one with the minimum $Z(t, p)$ value. □

Let π be the schedule obtained by the above approximation algorithm. Let Z and Z^* be the objective values of the schedule π and an optimal schedule π^*, respectively.

Theorem 4.1. *$Z \le 2Z^*$ and the bound is tight.*

Proof. Let A^* and R^* be the sets of the accepted jobs and the rejected jobs in π^*, respectively. Let $r^* = \max\{r_j : J_j \in A^*\}$ and $p^* = \max\{p_j : J_j \in A^*\}$. By the definitions of r^* and p^*, we have $S_2(r^*, p^*) = \{J_j : r_j > r^*$ or $p_j > p^*\} \subseteq R^*$. Then, we have $Z^* \ge \max\{r^*, p^*\} + w(S_2(r^*, p^*))$. Thus, we have

$$Z \le Z(r^*, p^*) = r^* + p^* + w(S_2(r^*, p^*)) \le 2Z^*.$$

To show that the bound is tight, we consider the following instance with 2 jobs: $(r_1, p_1, w_1) = (0, 1, 2)$, $(r_2, p_2, w_2) = (1, 0, 2)$. It is easy to verify that $Z(0, 0) = w_1 + w_2 = 4$, $Z(0, 1) = Z(1, 0) = 3$ and $Z = Z(1, 1) = 2$. However, the optimal schedule accepts both J_1 and J_2 with J_1 starting its processing at time 0 followed by J_2. That is, $Z^* = 1$. Thus, we have $Z = 2 = 2Z^*$. □

4.2. A fully polynomial-time approximation scheme

Let Z be the objective value of the above approximation algorithm. Let Z^* be the optimal objective value. By Theorem 4.1, we have $Z^* \le Z \le 2Z^*$. For any job J_j with $w_j > Z$, it is easy to see that $J_j \in A^*$. Otherwise, we have $Z^* \ge w_j > Z \ge Z^*$, a contradiction. If we modify the rejection penalty of such a job J_j by setting $w_i = Z$, the optimal objective value does not change. Thus, without loss of generality, we can assume that $w_j \le Z$ for $j = 1, \dots, n$. Now, we propose a fully polynomial-time approximation scheme A_ϵ for this problem.

Fully polynomial-time approximation scheme A_ϵ

Step 1. For any $\epsilon > 0$, set $M = \frac{\epsilon Z}{2n}$. Given an instance I, we define a new instance I' by rounding the rejection penalty of the jobs in I such that $w'_j = \lfloor \frac{w_j}{M} \rfloor M$, for $j = 1, \dots, n$.

Step 2. Apply the dynamic programming algorithm DPA to instance I' to obtain an optimal schedule $\pi^*(I')$ for instance I'.

Step 3. Replace the modified rejection penalty w'_j by the original rejection penalty w_j in $\pi^*(I')$ for each $j = 1, \dots, n$ to obtain a feasible solution π for instance I. □

Let Z_ϵ be the objective value of schedule π. We have the following theorem.

Theorem 4.2. *$Z_\epsilon \le (1 + \epsilon)Z^*$ and the running time of A_ϵ is $O(\frac{n^4}{\epsilon})$.*

Proof. Let $Z^*(I')$ be the optimal objective value of schedule $\pi^*(I')$. From Step 1 of A_ϵ, we have $w'_j \le w_j < w'_j + M$, and so, $Z^*(I') \le Z^*$. Replace w'_j by w_j for each $j = 1, \dots, n$, we have

$$Z_\epsilon \le Z^*(I') + \sum_{j=1}^{n} (w_j - w'_j) \le Z^* + nM \le Z^* + \frac{\epsilon Z}{2} \le (1 + \epsilon)Z^*.$$

Since $w_j \leq Z$ for $j = 1, \ldots, n$, we have $\sum_{j=1}^{n} \lfloor \frac{w_j}{M} \rfloor \leq \frac{2n}{\epsilon} \sum_{j=1}^{n} \frac{w_j}{Z} \leq \frac{2n^2}{\epsilon}$. Note that $w'_j = \lfloor \frac{w_j}{M} \rfloor M$ for $j = 1, \ldots, n$. Then, in the dynamic programming algorithm, we have $W \in \{kM : 0 \leq k \leq \sum_{j=1}^{n} \lfloor \frac{w_j}{M} \rfloor\}$. That is, there are at most $\sum_{j=1}^{n} \lfloor \frac{w_j}{M} \rfloor = O(\frac{n^2}{\epsilon})$ choices for each W in DPA. So, the running time of DPA (also A_ϵ) is $O(\frac{n^4}{\epsilon})$. Theorem 4.2 follows. □

References

[1] P. Brucker, A. Gladky, H. Hoogeveen, M.Y. Kovalyov, C.N. Potts, S.L. van de Velde, Scheduling a batching machine, Journal of Scheduling 1 (1998) 31–54.

[2] P. Brucker, S. Knust, Complexity results for scheduling problem. http://www.mathematik.uniosnabrueck.de/reseach/OR/class/2007.

[3] Y. Bartal, S. Leonardi, A.M. Spaccamela, J. Sgall, L. Stougie, Multiprocessor scheduling with rejection, SIAM Journal on Discrete Mathematics 13 (2000) 64–78.

[4] T.C.E. Cheng, Z.H. Liu, W.C. Yu, Scheduling jobs with release dates and deadlines on a batching processing machine, IIE Transactions 33 (2001) 685–690.

[5] X. Deng, Y.Z. Zhang, Minimizing mean response time for batch processing systems, Lecture Notes on Computer Science 1627 (1999) 231–240.

[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] R.L. Graham, E.L. Lawer, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: A survey, Annals of Discrete Mathematics 5 (1979) 1–15.

[10] H. Hoogeveen, M. Skutella, G.J. Woeginger, Preemptive scheduling with rejection, Mathematics Programming 94 (2003) 361–374.

[11] C.-Y. Lee, R. Uzsoy, L.A. Martin-Vega, Efficient algorithms for scheduling semiconductor burn-in operations, Operations Research 40 (1992) 764–775.

[12] C.-Y. Lee, R. Uzsoy, Minimizing makespan on a single batch processing machine with dynamic job arrivals, International Journal of Production Research 37 (1999) 219–236.

[13] Z.H. Liu, J.J. Yuan, T.C.E. Cheng, On scheduling an unbounded batch machine, Operations Research Letters 31 (2003) 42–48.

[14] S. Seiden, Preemptive multiprocessor scheduling with rejection, Theoretical Computer Science 262 (2001) 437–458.

[15] L.Q. Zhang, L.F. Lu, J.J. Yuan, Single machine with release date and rejection to minimize makespan, EJOR (submitted for publication).