Bounded parallel-batch scheduling on single and multi machines for deteriorating jobs

Cuixia Miao\textsuperscript{a,},*, Yuzhong Zhang\textsuperscript{b}, Zhigang Cao\textsuperscript{c}

\textsuperscript{a} School of Mathematical Sciences, Qufu Normal University, Qufu, Shandong 273165, People's Republic of China
\textsuperscript{b} School of Management, Qufu Normal University, Rizhao, Shandong 276826, People's Republic of China
\textsuperscript{c} Key Laboratory of Management Decision and Information Systems AMSS, CAS, Beijing 100190, People's Republic of China

\textbf{A R T I C L E   I N F O}

\textbf{Article history:}
Received 28 July 2010
Received in revised form 26 April 2011
Accepted 24 May 2011
Available online 27 May 2011
Communicated by A.A. Bertossi

\textbf{Keywords:}
Analysis of algorithms
Parallel-batch scheduling
deteriorating job
NP-hard
Fully polynomial time approximation scheme (FPTAS)

\textbf{A B S T R A C T}

We consider the bounded parallel-batch scheduling problem in which the processing time of a job is a simple linear function of its starting time. The objective is to minimize the makespan. When the jobs have identical release dates, we present an optimal algorithm for the single-machine problem and an fully polynomial-time approximation scheme for the parallel-machine problem. When the jobs have distinct release dates, we show that the single-machine problem is NP-hard and present an optimal algorithm for one special case.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Traditional scheduling problems assume that the processing times of jobs are constant. However, the processing times may change in the real world. Examples can be found in steel production and fire fighting, etc., where any delay in processing a task may increase its completion time. The reader is referred to Kunnathur and Gupta [1], Sundararaghavan and Kunnathur [2] for other examples.

Research on scheduling of jobs with simple linear deteriorating processing times may date back to Mosheiov [3] who first considered single machine scheduling under the simple linear deteriorating assumption. He showed that several problems with various objective functions are polynomially solvable. Cheng and Ding [4] studied the step-deteriorating model and gave a pseudo-polynomial time algorithm for minimizing makespan. Mosheiov [5], Chen [6], Ji and Cheng [7] considered the scheduling problems with deteriorating jobs on parallel machines.

Brucker et al. [8] defined that a parallel-batch machine is a machine that can process up to \( b \) jobs simultaneously as a batch, and the processing time of the batch is equal to the longest time of any job in the batch. All jobs contained in the same batch start and complete at the same time. Once processing of a batch is initiated, it cannot be interrupted and other jobs cannot be introduced into the batch until processing is completed. The parallel-batch scheduling is motivated by burn-in operations in semiconductor manufacturing. Lee et al. [9] provided a background description, Webster and Baker [10] presented an overview of algorithms and complexity results for scheduling batch processing machines. This processing system has been extensively studied in the last decade (Potts and Kovalyov [11]; Zhang and Cao [12]; Liu and Yu [13]; Lu et al. [14]). For the parallel-batch scheduling, there are two distinct models: the \textit{bounded model}, in which the bound \( B \) for each batch size is effective, i.e., \( b < n \), and the \textit{unbounded model}, in which there is effectively no limit on the size of batch, i.e., \( b \geq n \), where \( n \) denotes the number of jobs.
and \( b \) denotes the batch capacity. Potts and Kovalyov [11] gave survey on both of the two models. All the above-mentioned results concerning parallel-batch scheduling assume that the processing times are constants.

But job deterioration and parallel-batch processing coexist in many realistic scheduling situations. Examples can be found in the steel production. Qi et al. [15] considered the unbounded parallel-batch scheduling problem with deteriorating jobs on a single machine. They gave polynomial time algorithms for minimizing maximum cost, the number of tardy jobs, and the total weighted completion times and proved the NP-hardness for minimizing the weighted number of tardy jobs.

Our contributions: We study the bounded parallel-batch scheduling problem with deteriorating jobs in this paper. To the best of our knowledge, this type of problem has never been discussed. We consider not only the single-machine variant, but also the multiple-machine variant.

The rest of the paper is organized as follows. We describe the model and introduce some preliminaries in Section 2. In Section 3, we present an optimal algorithm for the single-machine problem and an FPTAS algorithm for the parallel-machine problem when the jobs arrive simultaneously. In Section 4, when the jobs arrive dynamically, we show that the single-machine problem is NP-hard and present an optimal algorithm for one special case. We conclude the paper and suggest some interesting topics for future research in the last section.

2. Problem description, notation and preliminaries

The problem considered in this paper can be formally described as follows: Given a set \( M = \{M_1, \ldots, M_m\} \) of \( m \) parallel machines and a set \( J = \{J_1, \ldots, J_n\} \) of \( n \) independent jobs. Each job \( J_j \) has a weight \( w_j \), a release date \( r_j \) and a deteriorating rate \( \alpha_j \). We assume, as Mosheiov [3], that the actual processing time of job \( J_j \) is \( p_j = \alpha_j t \), where \( t \) is the starting time of \( J_j \) in a schedule \( \pi \). Here, we assume \( \min\{r_j | j = 1, \ldots, n\} = t_0 > 0 \) (when \( t_0 = 0 \), the completion time of each job will be 0). The objective is to minimize \( C_{\max} = \max\{C_j | j = 1, \ldots, n\} \), where \( C_j \) denotes the completion time of job \( J_j \) in one given schedule.

For the given batch \( B \), we denote its deteriorating rate and release date by \( \alpha(B) \) and \( r(B) \), respectively, and \( C(B) \) is the completion time. Then \( \alpha(B) = \max\{\alpha_j | J_j \in B\} \), \( r(B) = \max\{r_j | J_j \in B\} \), and \( C(B) = (1 + \alpha(B))S(B) \), where \( S(B) \) is the starting time of batch \( B \). Using the 3-field notation of Graham et al. [16], we denote our problems as \( 1|B, r_j, \alpha_j|C_{\max} \) and \( P_m|B, p_j = \alpha_j t|C_{\max} \).

Before proceeding, we present the following lemma stated by Mosheiov [3].

**Lemma 1.** For the single machine scheduling problem \( 1|p_j = \alpha_j t|C_{\max} \) if \( \pi = \{J_{[1]}, J_{[2]}, \ldots, J_{[n]}\} \), the starting time of job \( J_{[1]} \) is \( t_0 \), then the makespan is \( C_{\max}(\pi) = t_0 \prod_{j=1}^{n}(1 + \alpha_{[j]}) \).

3. Identical release dates

In this section, we assume that all jobs are released at time \( t_0 (> 0) \).

3.1. Optimal algorithm for problem \( 1|B, p_j = \alpha_j t|C_{\max} \)

Zhang and Cao [12] showed that the Fully Batch Longest Processing Time (FBLPT) rule is optimal for \( 1|B|C_{\max} \). Lu et al. [14] also involved the FBLPT rule. For the bounded parallel-batch scheduling with deteriorating jobs, we obtain a similar rule in \( O(n\log n) \) time as follows:

**Algorithm FBLDR** (Fully Batch Longest Deteriorating Rate)

**Step 1.** Re-index jobs in non-increasing order of their deteriorating rates such that \( \alpha_1 \geq \cdots \geq \alpha_n \).

**Step 2.** Form batches by placing jobs \( J_{[b+1]} \) through \( J_{[(j+1)b]} \) together in the same batch, for \( j = 0, 1, \ldots, \lfloor \frac{n}{b} \rfloor \), where \( \lfloor \frac{n}{b} \rfloor \) denotes the largest integer smaller than \( \frac{n}{b} \).

**Step 3.** Schedule the batches in any arbitrary order.

The schedule contains at most \( \lfloor \frac{n}{b} \rfloor + 1 \) batches and all batches are full except possibly the last one.

**Theorem 1.** Algorithm FBLDR solves problem \( 1|B, p_j = \alpha_j t|C_{\max} \) optimally, and the optimal objective value is \( C_{\max} = t_0 \prod_{k=1}^{\lfloor \frac{n}{b} \rfloor}(1 + \alpha_{k+1}) \).

We omit the proof as it is simple.

3.2. An FPTAS for problem \( P_m|B, p_j = \alpha_j t|C_{\max} \)

In this subsection, we assume that \( t_0 \) and all \( \alpha_j \) for every \( j (1 \leq j \leq n) \) are integral. Ji and Cheng [7] showed that \( P_m|p_j = \alpha_j t|C_{\max} \) is NP-hard. So the following theorem is trivial.

**Theorem 2.** The problem \( P_m|B, p_j = \alpha_j t|C_{\max} \) is NP-hard.

Now, we are ready to give an FPTAS algorithm for this problem when \( m \) is fixed.

An algorithm \( A \) is called a \((1 + \varepsilon)\)-approximation algorithm for a minimization problem if it produces a solution that is at most \((1 + \varepsilon)\) times as big as the optimal value, running in time that is polynomial in the input size. A family approximation algorithms \( \{A_\varepsilon\} \) is a fully polynomial-time approximation scheme (FPTAS) if, for each \( \varepsilon > 0 \), the algorithm \( A_\varepsilon \) is a \((1 + \varepsilon)\)-approximation algorithm that is polynomial in the input size and in \( \frac{1}{\varepsilon} \). W.l.o.g., we assume that \( 0 < \varepsilon \leq 1 \).

We re-index jobs in non-increasing order of their deteriorating rates so that \( \alpha_1 \geq \cdots \geq \alpha_n \). In the above subsection, we showed that FBLDR is an optimal algorithm for the single machine, which leads to the following properties.

**Property 1.** For problem \( P_m|B, p_j = \alpha_j t|C_{\max} \) there exists an optimal schedule satisfying the following properties:

(i) The indices of jobs in each batch on every machine are consecutive;
(ii) All batches are full except possibly the one which contains job \( J_n \).

**Proof.** We consider any optimal schedule \( \pi = \{ \pi^1, \ldots, \pi^m \} \) on \( m \) identical parallel machines, where \( \pi^i = (B^i_1, \ldots, B^i_{n_i}) \) is the subsequence on machine \( M_i \) (\( i = 1, \ldots, m \)) in \( \pi \).

To show (i), consider two batches \( B^i_1 \) and \( B^i_2 \) perhaps on different machines, suppose there are three jobs \( J_j, J_{j+1}, J_{j+2} \) with \( \alpha_j \geq \alpha_{j+1} \geq \alpha_{j+2} \) and \( J_j, J_{j+2} \in B^i_1; J_{j+1} \in B^i_2 \).

Suppose we move \( J_{j+1} \) to \( B^i_1 \) and \( J_{j+2} \) to \( B^i_2 \), i.e., \( B^i_1 = B^i_1 \setminus \{J_{j+1}\} \cup \{J_{j+2}\}, B^i_2 = B^i_2 \setminus \{J_{j+2}\} \cup \{J_{j+1}\} \).

Since \( \alpha_j \geq \alpha_{j+1} \geq \alpha_{j+2} \), we have that \( \alpha(B^{i_1}_1) = \alpha(B^{i_1}_1), \alpha(B^{i_2}_2) \leq \alpha(B^{i_2}_2) \). Thus, the objective value does not increase after the swap whether \( l_1 = l_2 \) or not. Repeating this process, we see that in an optimal schedule, the indices of jobs in each batch on every machine are consecutive.

To show (ii), suppose there is a batch \( B_x \) in \( \pi \) such that \( B_x \) is not full. From (i), we know that the indices of jobs in \( B_x \) are consecutive, w.l.o.g., let \( B_x = \{J_{n_x}, J_{n_x+1}, \ldots, J_{n_x+k}\} \), then \( k + 1 < b \). If we move the remaining \( b - (k + 1) \) jobs \( \{J_{n_x+k+1}, \ldots, J_{n_x+b-1}\} \) from other batches to \( B_x \), this procedure cannot increase the objective value. A finite number of repetitions of this procedure yields an optimal schedule of the required form. \( \square \)

From Lemma 1, we get another property:

**Property 2.** For given \( k \) batches \( B_1, B_2, \ldots, B_k \), there exists an optimal batch sequence for \( P_m|B; p_j = \alpha_j t_j C_{\max} \) such that on each machine batches are sequence-independent.

For problem \( P_m|B; p_j = \alpha_j t_j C_{\max} \), the properties allow us to determine the batch structure of an optimal solution a priori. So we divide job into batches \( B_1, B_2, \ldots, B_k \) according to Algorithm FBLDR, where \( k = \left\lceil \frac{m}{2} \right\rceil \). It is possible to view the batch \( B_j \) (\( j = 1, \ldots, k \)) as single aggregate job with deteriorating rate \( \alpha(B_j) = \alpha(j-1)b+1 \).

Now, we introduce the variables \( x_j, x_{j} \in \{1, 2, \ldots, m\}, j = 1, 2, \ldots, k \), where \( x_j = l \) if batch \( B_j \) is scheduled on \( M_l \) (\( l = 1, 2, \ldots, m \)). Let \( X \) be the set of all vectors \( x = (x_1, x_2, \ldots, x_k) \) with \( x_j = l, j = 1, 2, \ldots, k, l = 1, 2, \ldots, m \).

Set \( F_0^j(x) = t_0 \quad i = 1, 2, \ldots, m \), \( F_1^j(x) = (1 + \alpha(B_j))F_{i-1}^j(x) \) for \( x_j = l \), \( F_{i-1}^j(x) = F_{i-1}^j(x) \) for \( x_j = l, i \neq 1 \).

Then problem \( P_m|B; p_j = \alpha_j t_j C_{\max} \) is reduced to the following problem:

**Minimize** \( Q(x) = \max \left\{ F_i^j(x) \mid i = 1, 2, \ldots, m \right\} \) for \( x \in X \).

We introduce the procedure (Partition \( A(F, \rho) \)) proposed by Kovalyov and Kubiak [17], where \( A \subseteq X, F \) is a nonnegative integer function on \( X \), and \( 0 < \rho \leq 1 \). This procedure partitions \( A \) into disjoint subsets \( A^F_1, A^F_2, \ldots, A^F_k \) such that \( |F(x) - F(x')| \leq \rho \min \{F(x), F(x')\} \) for any \( x, x' \in A^F_j \), \( j = 1, 2, \ldots, k_F \). The following description provides the details of Partition \( (A, F, \rho) \).

**Procedure Partition \( (A, F, \rho) \)**

Arrange the vectors \( x \in A \) in the order \( x^{(1)}, x^{(2)}, \ldots, x^{(|A|)} \), where \( 0 \leq F(x^{(1)}) \leq F(x^{(2)}) \leq \ldots \leq F(x^{(|A|)}) \). Assign the vectors \( x^{(1)}, x^{(2)}, \ldots, x^{(|A|)} \) to set \( A^F_1 \) until \( i_1 \) is found such that \( F(x^{(i_1)}) \leq (1 + \rho) F(x^{(1)}) \) and \( F(x^{(i+1)}) > (1 + \rho) F(x^{(i)}) \). If such \( i_1 \) does not exist, then take \( A^F_1 = A \) and stop.

Assign \( x^{(i_1+1)}, x^{(i_1+2)}, \ldots, x^{(i_2)} \) to set \( A^F_2 \) until \( i_2 \) is found such that \( F(x^{(i_2)}) \leq (1 + \rho) F(x^{(i_1+1)}) \) and \( F(x^{(i_2+1)}) > (1 + \rho) F(x^{(i_1+1)}) \). If such \( i_2 \) does not exist, then take \( A^F_2 = A - A^F_1 \) and stop.

Continue the above process until \( x^{(|A|)} \) is included in \( A^F_k \) for some \( k_F \).

The main properties of Partition were given by Kovalyov and Kubiak [17] as follows.

**Proposition 1.** \( |F(x) - F(x')| \leq \rho \min \{F(x), F(x')\} \) for any \( x, x' \in A^F_j \), \( j = 1, 2, \ldots, k_F \).

**Proposition 2.** \( k_F \leq \frac{\log F(x^{(|A|)})}{\rho} + 2 \) for \( 0 < \rho \leq 1 \) and \( 1 \leq F(x^{(|A|)}) \).

In the following, we give an fully polynomial time approximation scheme for problem \( P_m|B; p_j = \alpha_j t_j C_{\max} \) when \( m \) is fixed.

**Algorithm \( A^F \)**

**Step 1.** Re-index jobs in non-increasing order of their deteriorating rates so that \( \alpha_1 \geq \ldots \geq \alpha_n \).

**Step 2.** Form batches by placing jobs \( J_{j+1} \) through \( J_{j+1}b \) together in the same batch \( B_{j+1} \) for \( j = 0, 1, \ldots, \left\lfloor \frac{m}{b} \right\rfloor \).

**Step 3.** Regard batch \( B_{j+1} \) as an aggregate job with deteriorating rate \( \alpha(B_{j+1}) = \alpha_{j+1} \) for \( j = 0, 1, \ldots, \left\lfloor \frac{m}{b} \right\rfloor \).

**Step 4.** Set \( Y_0 = \{0, 0, \ldots, 0\} \), \( i = 1 \) and \( F^i_j(x) = t_0 \) for \( i = 1, 2, \ldots, m \).

**Step 5.** For the set \( Y_{j-1} \), generate the set \( Y'_j \) by adding \( l (l = 1, 2, \ldots, m) \) in position \( j \) of each vector from \( Y_{j-1} \).

Calculate the following for any \( x \in Y'_j \), w.l.o.g., assuming \( x_j = l \).

\( F^i_j(x) = (1 + \alpha(B_j))F^i_{j-1}(x), \) \( F^i_j(x) = F^i_{j-1}(x) \) for \( i \neq l \).

If \( j = k \), then set \( Y_k = Y'_k \), and go to Step 6.

If \( j < k \), then set \( \rho = \frac{\rho}{\rho + 1} \), and perform the following computation.

Call Partition \( (Y'_j, F^i_j, \rho) \) \((l = 1, \ldots, m)\) to partition the set \( Y'_j \) into disjoint subsets \( Y^i_{j,1}, Y^i_{j,2}, \ldots, Y^i_{j,k'} \).
Divide set $Y_j$ into disjoint subsets $Y_{a_1 \cdots a_m} = Y_{a_1} \cap \cdots \cap Y_{a_m}$, $a_1 = 1, \ldots, k_{F_1}; \ldots; a_m = 1, \ldots, k_{F_m}$. For each nonempty subset $Y_{a_1 \cdots a_m}$, choose a vector $x_{a_1 \cdots a_m}$ such that 
\[
F_j'(x_{a_1 \cdots a_m}) = \min \left\{ \max_{i=1, \ldots, m} F_j(x) \mid x \in Y_{a_1 \cdots a_m} \right\}.
\]

Set $Y_j := \{x_{a_1 \cdots a_m} \mid a_1 = 1, \ldots, k_{F_1}; \ldots; a_m = 1, \ldots, k_{F_m} \}$, and $Y_j' \cap \cdots \cap Y_{a_m} \neq \phi$, and $j = j + 1$. Repeat Step 5.

**Step 6.** Select set $x^0 \in Y_k$ such that $Q(x^0) = \min_{x \in Y_k} \{ \max_{i=1, \ldots, m} F_j'(x) \}$.

Let $L = \log(\max\{k, \frac{1}{\epsilon}, 1 + \alpha_{\max}, t_0\})$, where $\alpha_{\max} = \max(\alpha_j \mid j = 1, 2, \ldots, n)$ and $k = \lceil \frac{2}{\epsilon} \rceil$ denotes the smallest integer larger than or equal to $\frac{2}{\epsilon}$.

**Analysis of computational complexity:**

By Algorithm $A^B$, we note that Step 5 requires the time of $O(|Y_j||\log Y_j|)$ to complete, and we have $|Y_j'| \leq 2|Y_j| \leq 2k_{F_1} \cdots k_{F_m}$.

By Proposition 2 of Partition, for $i = 1, \ldots, m$, 
\[
k_{F_i} \leq 2\left(\lceil \frac{2}{\epsilon} \rceil + 1\right) \log(t_0(1 + \alpha_{\max})^{\lceil \frac{2}{\epsilon} \rceil}) + 2 \\
\leq \frac{12L}{\epsilon} + 2.
\]

So $|Y_j| = O\left(\left(\frac{2}{\epsilon}\right)^{2m+1} \frac{m}{\epsilon} \right)$ and $|Y'_j| \log|Y'_j| = O\left(\left(\frac{2}{\epsilon}\right)^{2m+1} \frac{m}{\epsilon} \right)$.

Thus, we have the time complexity of algorithm $A^B$ is $O\left(\left(\frac{2}{\epsilon}\right)^{2m+1} \frac{m}{\epsilon} \right)$.

Similarly to Ji and Cheng [7], we get the following theorem with the proof omitted (see Ji and Cheng [18] for details).

**Theorem 3.** Algorithm $A^B$ finds $x^0 \in X$ for $P_m\{B, p, j = \alpha j, t\}$ such that $Q(x^0) \leq (1 + \epsilon)Q(x')$ in $O\left(\left(\frac{2}{\epsilon}\right)^{2m+1} \frac{m}{\epsilon} \right)$, where $x'$ is an optimal solution.

**4. Distinct release dates**

**4.1. NP-hardness proof**

**Theorem 5.** The problem $\{B, r_j \in \{t_0, t_1\}, p, j = \alpha j, t\}$ is NP-hard even when the batch capacity $b = 2$.

**Proof.** We use a reduction from the Subset-Product problem.

An instance $I$ of the Subset-Product problem is formulated as follows: Given positive integers $x_1, \ldots, x_m$ such that $\prod_{i=1}^m x_i = A^2$, does there exist a subset $N_1$ of set $N = \{1, \ldots, m\}$ such that $\prod_{i \in N_1} x_i = A$?

In the above instance, we can omit the element $i \in N$ with $x_i = 1$, because it will not affect the product of any subset. Therefore, w.o.l.g., we can assume that $x_i \geq 2$ for every $i \in N$.

For any given instance $I$ of the Subset-Product problem, we construct a corresponding instance $II$ of our problem as follows: there are $n = 4m$ jobs, for each type $i (1 \leq i \leq m)$, we define four jobs: $J_{i1}, J_{i2}, J_{i3}, J_{i4}$, their deteriorating rates and release dates are given by 
\[
\alpha_{i1} = A^{4i}x_i - 1, \quad r_{i1} = t_0 - 1, \\
\alpha_{i2} = A^{4i}x_i - 1, \quad r_{i2} = r_{i3} = t_0 = 1, \\
\alpha_{i4} = A^{4i} - 1, \quad r_{i4} = A^{4(m+1)} - 1.
\]

Let $G = A^{4m(m+1)}$ be the threshold value.

It is clear that the reduction can be done in polynomial time, and it is easy to verify that $\alpha_{ij} > 0$ for $i = 1, \ldots, m, j = 1, \ldots, 4$.

We will prove that the instance $I$ has a solution if and only if the instance $II$ has a schedule $\pi$ with $C_{\max}(\pi) \leq G$.

To prove the necessity, suppose that Subset-Product problem has a solution. W.l.o.g., we assume that $N_1 = \{1, \ldots, k\}$, then $N\setminus N_1 = \{k+1, \ldots, m\} = N_2$ and $\prod_{i \in N_1} x_i = \prod_{i=k+1}^m x_i = A = \prod_{i=k+1}^m x_i$. Now, we construct the following schedule $\pi$:

\[
J_{11} \cdots J_{ik1}(k+1) \cdots J_{im1} \quad J_{12} \cdots J_{ik2}(k+1) \cdots J_{im2} \quad J_{14} \cdots J_{ik4}(k+1) \cdots J_{im4},
\]

where the two jobs in the same column are processed as a batch.

Because $t_0 \prod_{i=k+1}^m (\alpha_{i1} + 1) \prod_{i=k+1}^m (\alpha_{i2} + 1) = A^{2m(m+1)} = r_{i4}$, so the schedule $\pi$ is feasible. Then we have

\[
C_{\max}(\pi) = t_0 \prod_{i=1}^k (\alpha_{i1} + 1) \prod_{i=k+1}^m (\alpha_{i2} + 1) \\
\times \prod_{i=1}^k (\alpha_{i4} + 1) \prod_{i=k+1}^m (\alpha_{i1} + 1) \\
= t_0 \prod_{i=1}^k A^{4i}x_i \prod_{i=k+1}^m A^{4i}x_i \\
\times \prod_{i=1}^k A^{4i}x_i \prod_{i=k+1}^m A^{4i}x_i \\
= \left( \prod_{i=1}^k A^{4i}x_i \right)^2 \\
= A^{4m(m+1)+1} \\
= G.
\]

To prove the sufficiency, suppose that there is a schedule $\pi$ satisfies $C_{\max}(\pi) \leq G = A^{4m(m+1)+1}$.

To show the instance $I$ has a solution, we will prove the following observation, firstly.

**Observation 1.** $\pi$ is a schedule with $C_{\max}(\pi) \leq G$. Then every batch in $\pi$ contains two jobs of the same type.

**Proof.** Let $E(D)$ be defined for each batch $D \in \pi$ as follows:

- If $|D| = 1$, we define $E(D) = (\alpha_D) |J_{ij} \in D| + 1$.
- If $|D| = 2$, we define $E(D) = \max(\alpha_D |J_{ij} \in D| + 1)$.

Let the deteriorating rate of batch $D$ be $\alpha(D)$, then we have $\alpha(D) + 1 = \sqrt{\prod_{i \in D} (\alpha_{ij} + 1) E(D)}$. For all the batches $D \in \pi$, we get
Because $C_{\max}(\pi) \leq G = A^{4(m+1)+1}$, from the above equation, we have $\prod_{D \in \pi} E(D) \leq A^4$.

If a batch $D$ contains two jobs in distinct types or only a single job, then we get $E(D) > A^2$. Thus, we get that $\pi$ cannot have two or more such batches. On the other hand, as we know, the total number of jobs is even and the number of jobs in each type is even, thus only one such batch in $\pi$ is not feasible. Therefore, every batch in $\pi$ contains two jobs of the same type. This completes the Observation 1.

We are ready to prove that instance $I$ has a solution.

From the above observation, we know that every batch in $\pi$ contains two jobs of the same type. Since $J_{12}$ is identical with $J_{13}$, there are two ways to partition four types of job $i$ into two batches $\{(J_{11}, J_{44})\}$ or $\{(J_{11}, J_{12}); (J_{13}, J_{44})\}$. W.l.o.g., let batches be $\{(J_{11}, J_{12}); (J_{13}, J_{44})\}$ for $i = 1, \ldots, k$, and $\{(J_{11}, J_{12}); (J_{13}, J_{44})\}$ for $i = k + 1, \ldots, m$.

As $r_{11} = r_{12} = r_{13} = 1$ and $r_{14} = A^{2m(m+1)}$, the following schedule $\pi$ is feasible:

$\begin{align*}
J_{11} & \ldots J_{1k} J_{(k-1), 3} \ldots J_{m3} J_{13} \ldots J_{k3} J_{(k+1), 1} \ldots J_{m1}, \\
J_{12} & \ldots J_{1k} J_{(k+1), 2} \ldots J_{m2} J_{14} \ldots J_{k4} J_{(k+1), 4} \ldots J_{m4},
\end{align*}$

where the two jobs in the same column are processed as a batch. Thus, we have

$C_{\max}(\pi) = \max\left\{ \sum_{i=1}^{k} \left( \frac{1}{\alpha_{i1} + 1} \right) \prod_{i=k+1}^{m} \left( \frac{\alpha_{i2} + 1}{\alpha_{i2} + 1} \right), r_{14} \right\}$

$= \max\left\{ \left( \prod_{i=1}^{k} A^{\alpha_{i1} + 1} \prod_{i=k+1}^{m} A^{\alpha_{i2} + 1} \right), r_{14} \right\}$

$= \max\left\{ \sum_{i=1}^{k} \left( A^{\alpha_{i1} + 1} \right) \prod_{i=k+1}^{m} A^{\alpha_{i2} + 1} \right\}$

$= \max\left\{ \sum_{i=1}^{m} \left( A^{\alpha_{i1} + 1} \right) \prod_{i=k+1}^{m} A^{\alpha_{i2} + 1} \right\}$

$= \max\left\{ \sum_{i=1}^{m} \left( A^{\alpha_{i1} + 1} \right) \prod_{i=k+1}^{m} A^{\alpha_{i2} + 1} \right\}$

$= \max\left\{ \sum_{i=1}^{m} \left( A^{\alpha_{i1} + 1} \right) \prod_{i=k+1}^{m} A^{\alpha_{i2} + 1} \right\}$

Thus, it holds $\prod_{i=1}^{m} x_i \prod_{i=k+1}^{m} x_i = A^2$. Thus, it holds

$\prod_{i=1}^{m} x_i \prod_{i=k+1}^{m} x_i = A^2$. Thus, it holds

$\prod_{i=1}^{m} x_i \prod_{i=k+1}^{m} x_i = A^2$. Thus, it holds

As $C_{\max}(\pi) \leq G = A^{4(m+1)+1}$, we have

$\max\left\{ \sum_{i=1}^{k} \left( A^{\alpha_{i1} + 1} \right) \prod_{i=k+1}^{m} A^{\alpha_{i2} + 1} \right\} \leq A$.

Note that $\prod_{i=1}^{m} x_i \prod_{i=k+1}^{m} x_i = A^2$. Thus, it holds

As $C_{\max}(\pi) \leq G = A^{4(m+1)+1}$, we have

$\max\left\{ \sum_{i=1}^{k} \left( A^{\alpha_{i1} + 1} \right) \prod_{i=k+1}^{m} A^{\alpha_{i2} + 1} \right\} \leq A$.

Note that $\prod_{i=1}^{m} x_i \prod_{i=k+1}^{m} x_i = A^2$. Thus, it holds

As $C_{\max}(\pi) \leq G = A^{4(m+1)+1}$, we have

$\max\left\{ \sum_{i=1}^{k} \left( A^{\alpha_{i1} + 1} \right) \prod_{i=k+1}^{m} A^{\alpha_{i2} + 1} \right\} \leq A$.

Note that $\prod_{i=1}^{m} x_i \prod_{i=k+1}^{m} x_i = A^2$. Thus, it holds

As $C_{\max}(\pi) \leq G = A^{4(m+1)+1}$, we have

$\max\left\{ \sum_{i=1}^{k} \left( A^{\alpha_{i1} + 1} \right) \prod_{i=k+1}^{m} A^{\alpha_{i2} + 1} \right\} \leq A$.

Note that $\prod_{i=1}^{m} x_i \prod_{i=k+1}^{m} x_i = A^2$. Thus, it holds
5. Conclusion

In this paper, we considered the bounded parallel-batch scheduling with deteriorating jobs to minimize the makespan. We present an optimal algorithm for the single-machine problem and an FPTAS algorithm for the parallel-machine problem when the jobs arrive simultaneously. When the jobs arrive dynamically, we show that the single machine problem is NP-hard and present an optimal algorithm for one special case. For future research, it is worth considering other objectives.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 11071142), the PH.D. Foundation of Shandong Province (No. 2007BS01014) and the Natural Science Foundation of Shandong Province (No. ZR2010AM034).

References