MULTI-MACHINE SCHEDULING WITH INTERVAL CONSTRANGED POSITION-DEPENDENT PROCESSING TIMES

XIANYU YU
College of Economics and Management
Nanjing University of Aeronautics and Astronautics
Nanjing, 211106, China

DAR-LI YANG
Department of Information Management,
National Formosa University, YunLin 63201, Taiwan

DEQUN ZHOU AND PENG ZHOU
College of Economics and Management
Nanjing University of Aeronautics and Astronautics
Nanjing, 211106, China

(Communicated by Ada Che)

Abstract. This paper investigates multi-machine scheduling problems with interval constrained actual processing times. The actual processing time of each job is assumed to be restricted in a given interval otherwise the extra earliness or tardiness time should be used to patch up the flaw of job. The objectives are to find the optimal job sequence to minimize the total load of machines, the number of exceeding-interval jobs and the makespan of job schedule, respectively. This paper shows that both of the total load minimization problem and the exceeding job number minimization problem are polynomially solvable. For the makespan minimization problem, this paper proves that it is NP-hard, and proposes a fully polynomial time approximation scheme (FPTAS) for the case with two parallel machines.

1. Introduction. In the steel rolling mill, the temperature of an ingot while waiting to enter the rolling machine drops below a certain level, requiring the ingot to be reheated before rolling in Gupta and Gupta [10]. Motivating from the variable processing times in various effects such as the reheat requirement of ingots in steel rolling mills, many scheduling literatures paid attention to the scheduling problems in which the processing time of a job is not constant. References Gupta and Gupta [10] and Browne and Yechiali [3] introduced the variable job processing times due to deteriorating effects into scheduling studies, and proposed the optimal sequencing principle of minimizing the makespan. The total flow time minimization problem with linear actual processing times of jobs was considered by Mosheiov [16], and its optimal schedule was proved to be V-shaped of basic processing times of jobs.

2010 Mathematics Subject Classification. Primary: 90B35.
Key words and phrases. Scheduling, total load, makespan, NP-hard, FPTAS.

The first author is supported by the Fundamental Research Fund for the Central Universities No. XAA16077.

* Corresponding author: Dar-Li Yang.
Based on general job-dependent learning curves, Mosheiov and Sidney [18] established several models of job processing times and showed that two single machine problems and two multi-machine problems remain polynomially solvable. Liu et al. [15] studied parallel-machine scheduling problem with linear deteriorating effect of minimizing the makespan of jobs, and construct an FPTAS to solve the problem, which shows that the considered problem is NP-hard in the ordinary sense. Many other scheduling studies with variable processing times of jobs can be found in Biskup [1], Kuo and Yang [14], Mosheiov [17], Ji et al. [13], Yin et al. [28], Zhang et al. [31], etc.

Another interesting topic is scheduling with interval constraints of processing times. Chen et al. [4] explored to propose the algorithm for generating optimal cyclic schedules of hoist where the time of a printed circuit board spend in a tank is upper and lower bounded. Yan et al. [27] proposed a branch and bound algorithm for optimal cyclic scheduling in a robotic cell where the exact processing time of each part is limited in a time window. Several recent papers (see, e.g., Zhou et al. [32], Yan et al. [26] and Yan et al. [25]) also paid their attention to explore the cyclic robotic cell scheduling problems with time constraints. To cope with the quality requirement of job processing, the actual processing times of jobs are required to be fallen in given intervals in several recent studies, else if the processing time of a job is not within its time interval may lead to quality flaws, the extra cost or repair time should be paid. Yu and Zhang [29] studied single-machine scheduling problems with general position-dependent aging effect and machine maintenance, and assumed that the actual processing times of jobs should not exceed the upper-bounds, and proved that the considered problems remain polynomial solvable. Xue et al. [24] considered the single machine scheduling problem with machine maintenance and interval constrained processing times, where each job has a processing interval, the optimal polynomial time algorithms are designed to the considered problem. Finke et al. [8] showed the single machine problem with machine maintenance and interval constraints of processing times remains polynomial solvable if the objective is extended from the makespan of schedule to another four criterions.

To the best of our knowledge, it is rare to see the research of multi-machine scheduling with both general position-dependent deteriorating effect and interval constraints of job processing times, which is the common scenario in the actual production process. For an example, in the porcelain firing workshop with multiple furnaces, the actual processing time of porcelain on each furnace is in deteriorating effect due to the wear of furnace, the firing time of each porcelain batch should be in a given time interval to guarantee the quality of porcelain. Three objectives are considered in this paper. Firstly, in order to minimize the cost of machine running, minimizing the total processing times on machines (i.e., the total load of machines) is the first considered objective. Second, the number of jobs exceeding the time interval should be minimized for the reason that the processing time of job exceeding the time interval lead to the bad quality product. The third considered objective is to minimize the makespan of schedule to obtain the most efficient schedule.

This paper is organized as follows. In section 2, the problem formulation is introduced. In section 3, the total load minimization problem and the exceeding job number minimization problem are considered. In section 4, we propose the NP-hard analysis for the makespan minimization problem, and design an FPTAS to solve a special case. Finally, we conclude the paper.
2. **Notation and problem formulation.** The notation and terminology throughout this paper is proposed as follows. There are \( n \) independent jobs ready to be processed on \( m \) machines at time zero. Preemption is not allowed and a machine is only able to process one job at a time. The number of jobs scheduled on machine \( i \) is denoted as \( n_i \), which can be inferred to satisfy

\[
\sum_{i=1}^{m} n_i = n, \quad (i = 1, 2, \ldots, m) .
\]  

Job \( j \) has a normal processing time \( p_j \) \((j = 1, 2, \ldots, n)\). If job \( j \) is assigned to the \( r \)th position on machine \( i \), its the actual processing time \( p_{ij}^r \) is equal to

\[
f_{ij}(i,j,r), \quad (1 \leq i \leq m, 1 \leq j \leq n, 1 \leq r \leq n_i),
\]  

where \( f_{ij}(i,j,r) \) is not restricted to be specific and monotone, and can vary with machine \( i \) and job \( j \). When job \( j \) is assigned to the first position on a machine, its actual processing time is equal to its normal processing time, i.e., \( f_{ij}(i,j,1) = p_j \). The actual processing time of job \( j \) is required in the time interval \([a_j, b_j]\). If the actual processing time of a job exceed its corresponding interval, the repair time should be added to restore the quality of the job. Concretely, the earliness repair time of job \( j \) according to the time interval \([a_j, b_j]\) is denoted as

\[
E_j = \alpha_1 \max(0, a_j - f_{ij}(i,j,r)),
\]  

and the tardiness repair time of job \( j \) according to the time interval \([a_j, b_j]\) is denoted as

\[
T_j = \alpha_2 \max(0, f_{ij}(i,j,r) - b_j),
\]  

where \( \alpha_1 \) and \( \alpha_2 \) are the parameters of the tardiness repair time and the tardiness repair time, respectively. The normal processing time \( p_j \) is assumed to fall in the interval \([a_j, b_j]\), i.e., \( a_j \leq p_j \leq b_j \).

The completion time of job \( j \) on machine \( i \) is denoted by \( C_{ij} \). The total load on machines can be denoted as

\[
TL = \sum_{i=1}^{m} \max\{C_{ij}, j = 1, 2, \ldots, n_i\} = \sum_{j=1}^{n} (p_{ij}^* + E_j + T_j) \]  

Based on (2), (3) and (4), the total load \( TL \) can be transferred as

\[
TL = \sum_{j=1}^{n} \sum_{i=1}^{m} \sum_{r=1}^{n_i} [f_{ij}(i,j,r) + \alpha_1 \max(0, a_j - f_{ij}(i,j,r))
+ \alpha_2 \max(0, f_{ij}(i,j,r) - b_j)] x_{ijr},
\]  

when job \( j \) is scheduled in the \( r \)th position on machine \( i \), \( x_{ijr} = 1 \), otherwise \( x_{ijr} = 0 \).

The number of job \( j \) exceeding the time interval \([a_j, b_j]\) is

\[
U_j = g_{ij}(i,j,r) = \begin{cases} 
0, & \text{if } a_j \leq f_{ij}(i,j,r) \leq b_j \\
1, & \text{if } f_{ij}(i,j,r) < a_j \text{ or } f_{ij}(i,j,r) > b_j.
\end{cases}
\]  

Based on (7), the number of all the exceeding intervals jobs can be denoted as

\[
\sum U_j = \sum_{j=1}^{n} \sum_{i=1}^{m} \sum_{r=1}^{n_i} g_{ij}(i,j,r)x_{ijr}
\]
when job \( j \) is scheduled in the \( r \)th position on machine \( i \), \( x_{ijr} = 1 \), otherwise \( x_{ijr} = 0 \).

The completion time of job \( j \) on machine \( i \) is denoted by \( C_{ij} \), and the last completion time of jobs on machine \( i \) is denoted by \( C_i \), i.e. \( C_i = \max\{C_{ij}, \ j = 1, 2, \ldots, n_i\} \). The makespan of the schedule is \( C_{\text{max}} = \max\{C_i, \ i = 1, 2, \ldots, m\} \). Based on (2), (3) and (4), the makespan can be transferred as

\[
C_{\text{max}} = \max_{i=1,\ldots,m} \left\{ \sum_{j=1}^{n_i} \left( \sum_{r=1}^{n_{ij}} \left[ f_{ij}(i, j, r) + \alpha_1 \max(0, a_j - f_{ij}(i, j, r)) \right] \right) x_{ijr} \right\}. \tag{9}
\]

Let \( IC \) denote the character of interval constraint. By using the three-field notation scheme for scheduling problems introduced by Graham [9], the total load minimization problem is denoted as \( R_m|p_{ij}^r = f_{ij}(i, j, r), IC|TL \), the problem of minimizing the the number of all the exceeding intervals jobs can be denoted as \( R_m|p_{ij}^r = f_{ij}(i, j, r), IC|\sum U_j \), and the problem of minimizing the the number of all the exceeding intervals jobs can be denoted as \( R_m|p_{ij}^r = f_{ij}(i, j, r), IC|C_{\text{max}} \).

3. Multi-machine scheduling problem of minimizing the total load. In this section, the problem \( R_m|p_{ij}^r = f_{ij}(i, j, r), IC|TL \) is firstly analyzed.

**Theorem 3.1.** The problem \( R_m|p_{ij}^r = f_{ij}(i, j, r), IC|TL \) can be optimally solved in \( O(n^{m+2}) \) time.

**Proof.** The two-step procedure in Mosheiov [17] can be used to optimally solve the problem \( R_m|p_{ij}^r = f_{ij}(i, j, r), IC|TL \). The first step is giving an allocation of the numbers of jobs on machines \( (n_1, n_2, \ldots, n_m) \). Based on \( \sum_{i=1}^{m} n_i = n \) and the results in Ji and Cheng [12], it can be obtained that the above upper bound for the number of different allocations for \( n \) jobs \( m \) machines is \( O(n^{m-1}) \).

In the second step, for a given allocation on machines \( (n_1, n_2, \ldots, n_m) \), we can transfer the problem \( R_m|p_{ij}^r = f_{ij}(i, j, r), IC|TL \) as a standard \( (n \times n) \) assignment problem as follows.

\[
\begin{align*}
\text{min} & \quad \sum_{j=1}^{n} \sum_{i=1}^{m} \sum_{r=1}^{n_{ij}} w_{ijr} x_{ijr} \\
\text{s.t.} & \quad \sum_{j=1}^{n} x_{ijr} = 1, \ i = 1, 2, \ldots, m, \ r = 1, 2, \ldots, n_i, \\
& \quad \sum_{i=1}^{m} \sum_{r=1}^{n_{ij}} x_{ijr} = 1, \ j = 1, 2, \ldots, n, \\
& \quad x_{ijr} = 0 \text{ or } 1, \ j = 1, 2, \ldots, n, \ i = 1, 2, \ldots, m, \ r = 1, 2, \ldots, n_i,
\end{align*}
\]

where job \( j \) is scheduled in the \( r \)th position on machine \( i \), \( x_{ijr} = 1 \), otherwise \( x_{ijr} = 0 \). From (6), the weight \( w_{ijr} \) can be obtained by \( w_{ijr} = f_{ij}(i, j, r) + \alpha_1 \max(0, a_j - f_{ij}(i, j, r)) + \alpha_2 \max(0, f_{ij}(i, j, r) - b_j) \). As shown in Munkres [19], the above standard \( (n \times n) \) assignment problem can be optimally solved in \( O(n^3) \) time.

Combining two steps, it can be seen that the total computational complexity of solving the problem \( R_m|p_{ij}^r = f_{ij}(i, j, r), IC|TL \) is \( O(n^{m+2}) \).

In what follows, three special cases for the problem \( R_m|p_{ij}^r = f_{ij}(i, j, r), IC|TL \) are considered. In the first special case, the actual processing time \( f_{ij}(i, j, r) \)
is non-decreasing in \( r \), and the considered problem can be denoted as \( R_m|p_{ij}^r = f_{ij}(i, j, r), IC, ND|TL \), where \( ND \) is used to denoted the non-decreasing monotonicity of \( f_{ij}(i, j, r) \). Note that \( a_j \leq p_j \leq b_j \) and \( f_{ij}(i, j, r) \geq p_j \) due to the monotonicity of \( f_{ij}(i, j, r) \), the considered time interval \([a_j, b_j]\) can be reduced to \([p_j, b_j]\). Then we only need to analyze the tardiness repair time \( T_j = \alpha_2 \max(0, f_{ij}(i, j, r) - b_j) \) while \( E_j = \alpha_1 \max(0, a_j - f_{ij}(i, j, r)) = 0 \). Then the total load in this case can be obtained

\[
TL = \sum_{j=1}^{n} \sum_{i=1}^{m} \sum_{r=1}^{n} [f_{ij}(i, j, r) + \alpha_2 \max(0, f_{ij}(i, j, r) - b_j)]x_{ijr},
\]

(10)

when job \( j \) is scheduled in the \( r \)th position on machine \( i \), \( x_{ijr} = 1 \), otherwise \( x_{ijr} = 0 \).

**Theorem 3.2.** The problem \( R_m|p_{ij}^r = f_{ij}(i, j, r), IC, ND|TL \) can be optimally solved in \( O(mn^3) \) time.

**Proof.** Because that \( f_{ij}(i, j, r) \) is non-decreasing in \( r \), there exists an optimal schedule that all the jobs are continuously scheduled as previous as possible on each machine. Then the optimal schedule of the problem \( R_m|p_{ij}^r = f_{ij}(i, j, r), IC, ND|TL \) can be obtained by solving a \((mn \times n)\) rectangle assignment problem as follows.

\[
\min \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{r=1}^{n} w_{ijr}x_{ijr}
\]

\[
s.t. \sum_{j=1}^{n} x_{ijr} = 1, \quad i = 1, 2, \ldots, m, \quad r = 1, 2, \ldots, n,
\]

\[
\sum_{i=1}^{m} \sum_{r=1}^{n} x_{ijr} = 1, \quad j = 1, 2, \ldots, n,
\]

\[
x_{ijr} = 0 or 1, \quad j = 1, 2, \ldots, n, \quad i = 1, 2, \ldots, m, \quad r = 1, 2, \ldots, n,
\]

where job \( j \) is scheduled in the \( r \)th position on machine \( i \), \( x_{ijr} = 1 \), otherwise \( x_{ijr} = 0 \). From (6), the weight \( w_{ijr} \) can be obtained by \( w_{ijr} = f_{ij}(i, j, r) + \alpha_2 \max(0, f_{ij}(i, j, r) - b_j) \). Based on the extended Munkres algorithm in Bourgeois and Lassalle [2], the above \((mn \times n)\) assignment problem can be optimally solved in \( O(mn^3) \) time. \( \square \)

For the second special case, the machine setting is parallel-machine. Then the actual processing time of job \( j \) can be denoted as \( p_{ij}^r = f_j(j, r) \), which is also assumed to be non-decreasing in \( r \). This special case can be denoted as \( P_m|p_{ij}^r = f_j(j, r), IC, ND|TL \), where the total load \( TL \) can be calculated by

\[
\sum_{j=1}^{n} \sum_{r=1}^{m} \sum_{i=1}^{n} [f_j(j, r) + \alpha_2 \max(0, f_j(j, r) - b_j)]x_{ijr},
\]

when job \( j \) is scheduled in the \( r \)th position on machine \( i \), \( x_{ijr} = 1 \), otherwise \( x_{ijr} = 0 \).

**Theorem 3.3.** The problem \( P_m|p_{ij}^r = f_j(j, r), IC, ND|TL \) can be optimally solved in \( O(n^3) \) time.

**Proof.** Since \( f_j(i, j, r) \) is non-decreasing in \( r \), it is easy to obtained that \( f_j(i, j, r) + \alpha_2 \max(0, f_j(j, r) - b_j) \) is also non-decreasing in \( r \). Due to the monotonicity of \( f_j(i, j, r) + \alpha_2 \max(0, f_j(j, r) - b_j) \) and the parallel-machine setting, we can use the method of changing job position in Yu et al. [30] to prove that the number of jobs on machine \( i \) satisfies a balance principle, i.e., \( \lfloor n/m \rfloor \leq n_i \leq \lfloor n/m \rfloor + 1, i = 1, 2, \ldots, m. \) It is easy to obtain the possible positions for job allocation in
the optimal schedule based on the balance principle. Then the problem \( P_m|p^r_j = f_j(i, r), IC, ND|TL \) can be transferred as an standard \( n \times n \) assignment problem, which can be optimally solved in \( O(n^3) \) time.

In the third special case, if job \( j \) is assigned to the \( r \)th position on a machine, its actual processing time is

\[
p^r_j = h(p_j)l(r), \quad (1 \leq j \leq n, \ 1 \leq r \leq n),
\]

where the functions \( h(p_j) \) and \( l(r) \) are non-decreasing in \( p_j \) and non-decreasing in \( r \), respectively. The actual processing time of job \( j \) should be constrained in the time interval \([p_j, h(p_j)b_0] \) otherwise the extra repair time \( \alpha_2 \max(0, h(p_j)(l(r) - b_0)) \) should be added. This special case can be denoted by \( P_m|p^r_j = h(p_j)l(r), IC, ND|TL \). Similar with the two previous special cases, the total load of this case can be calculated as follows:

\[
TL = \sum_{j=1}^{n} \sum_{i=1}^{m} \sum_{r=1}^{n_i} [h(p_j)l(r) + \alpha_2 \max(0, h(p_j)(l(r) - b_0))]x_{ijr},
\]

when job \( j \) is scheduled in the \( r \)th position on machine \( i \), \( x_{ijr} = 1 \), otherwise \( x_{ijr} = 0 \). Before analyzing the solving algorithm of the problem \( P_m|p^r_j = h(p_j)l(r), IC, ND|TL \), we firstly introduce a useful lemma.

**Lemma 3.4.** (see, e.g., Hardy et al. [11]) If there are two sequences of numbers \( x_i \) and \( y_i \), then the sum of the products of the corresponding elements \( \sum x_iy_i \) is the least if the sequences are monotonic in the opposite sense.

**Theorem 3.5.** The problem \( P_m|p^r_j = h(p_j)l(r), IC, ND|TL \) can be optimally solved in \( O(n \log n) \) time.

**Proof.** Since \( p^r_j = h(p_j)l(r) \) is a specific form of \( p^r_j = f_j(i, r) \), the problem \( P_m|p^r_j = h(p_j)l(r), IC, ND|TL \) is a special case of the problem \( P_m|p^r_j = f_j(i, r), IC, ND|TL \). Based on the analysis of the problem \( P_m|p^r_j = f_j(i, r), IC, ND|TL \), it can be obtained that the number of jobs on machine \( i \) satisfies a balance principle, i.e., \([n/m] \leq n_i \leq [n/m] + 1, i = 1, 2, \ldots, m \) for the problem \( P_m|p^r_j = h(p_j)l(r), IC, ND|TL \). Then, the possible positions for job allocation in an optimal schedule can be proposed by the balance principle.

Since \( l(r) \) is non-decreasing in \( r \), it is easy to get that \( l(r) + \alpha_2 \max(0, (l(r) - b_0)) \) is also non-decreasing in \( r \). Because of (12), the total load for this considered case can be calculated as follows:

\[
TL = \sum_{j=1}^{n} \sum_{i=1}^{m} \sum_{r=1}^{n_i} h(p_j)[l(r) + \alpha_2 \max(0, (l(r) - b_0))]x_{ijr},
\]

when job \( j \) is scheduled in the \( r \)th position on machine \( i \), \( x_{ijr} = 1 \), otherwise \( x_{ijr} = 0 \). Due to Lemma 1 and the monotonicity of \( h(p_j) \) and \( l(r) \), an optimal schedule of the problem \( P_m|p^r_j = h(p_j)l(r), IC, ND|TL \) can be constructed by sequencing \( p_j \) and \( l(r) + \alpha_2 \max(0, (l(r) - b_0)) \) in the opposite monotonicity in \( O(n \log n) \) time.

4. **Multi-machine scheduling problem of minimizing the number of exceeding jobs.** In the section, the exceeding-job-number minimization problem \( R_m|p^r_{ij} = f_{ij}(i, j, r), IC|\sum U_j \) will be analyzed. Moreover, we will consider its three special cases based on the similar assumptions for \( R_m|p^r_{ij} = f_{ij}(i, j, r), IC \),
\( ND|TL, P_m|p_j^r = f_j(j, r), IC, ND|TL \) and \( P_m|p_j^r = h(p_j)l(r), IC, ND|TL \), respectively. In turn, the considered problems in this section can be denoted as \( R_m|p_j^r = f_j(i, j, r), IC, ND|\sum U_j, P_m|p_j^r = f_j(j, r), IC, ND|\sum U_j \) and \( P_m|p_j^r = h(p_j)l(r), IC, ND|\sum U_j \), respectively.

**Theorem 4.1.** The problem \( R_m|p_j^r = f_j(i, j, r), IC|\sum U_j \) can be optimally solved in \( O(nm^2) \) time.

**Proof.** After changing the weight \( w_{ijr} \) from \( f_j(i, j, r) + \alpha_1 \max(0, a_j - f_j(i, j, r)) + \alpha_2 \max(0, f_j(i, j, r) - b_j) \) to \( g_j(i, j, r) \), the two-step method in the proof of Theorem 3.1 can be used to solve the problem \( R_m|p_j^r = f_j(i, j, r), IC|\sum U_j \) in \( O(nm^2) \) time. \( \Box \)

Similarly, by changing the weight \( w_{ijr} \) from \( f_j(i, j, r) + \alpha_2 \max(0, f_j(i, j, r) - b_j) \) to \( g_j(i, j, r) \), we can use the proof method in Theorem 3.2 to obtain the following theorem.

**Theorem 4.2.** The problem \( R_m|p_j^r = f_j(i, j, r), IC, ND|\sum U_j \) can be optimally solved in \( O(mn^3) \) time.

When the machine setting is parallel machine, the number of job \( j \) exceeding the time interval \([a_j, b_j] \) is

\[
U_j = g_j(j, r) = \begin{cases} 
0 & \text{if } a_j \leq f_j(j, r) \leq b_j \\
1 & \text{if } f_j(j, r) < a_j \text{ or } f_j(j, r) > b_j. 
\end{cases}
\] (14)

Then the number of exceeding-interval jobs can be denoted as \( \sum U_j = \sum_{j=1}^{n} \sum_{i=1}^{m} \sum_{r=1}^{n} g_j(j, r) x_{ijr} \) when job \( j \) is scheduled in the \( r \)th position on machine \( i \), \( x_{ijr} = 1 \), otherwise \( x_{ijr} = 0 \). Similarly, after changing the weight \( w_{ijr} \) from \( f_j(j, r) + \alpha_2 \max(0, f_j(j, r) - b_j) \) to \( g_j(j, r) \), we can use the proof method in Theorem 3 to obtain the following theorem.

**Theorem 4.3.** The problem \( P_m|p_j^r = f_j(j, r), IC, ND|\sum U_j \) can be optimally solved in \( O(n^3) \) time.

Comparing with the problem \( P_m|p_j^r = h(p_j)l(r), IC, ND|TL \), the following theorem shows that the computational complexity of solving \( P_m|p_j^r = h(p_j)l(r), IC, ND|\sum U_j \) can be reduced from \( O(n \log n) \) to \( O(n) \).

**Theorem 4.4.** The problem \( P_m|p_j^r = h(p_j)l(r), IC, ND|\sum U_j \) can be optimally solved in \( O(n) \) time.

**Proof.** For the problem \( P_m|p_j^r = h(p_j)l(r), IC, ND|\sum U_j \), the time interval of the actual processing time of job \( j \) is \([p_j, h(p_j)l_0] \), the number of job \( j \) exceeding the time interval \([p_j, h(p_j)l_0] \) can be obtained by comparing \( l(r) \) and \( b_0 \), and then the number of all the exceeding-interval jobs is the number of jobs with \( l(r) > b_0 \). Since \( l(r) \) is non-decreasing in \( r \), we have \( \sum U_j = n - r_0 \), where \( r_0 = \min \{ r \mid l(r) > b_0 \} \). Obviously, the computational complexity is at most \( n \) times comparing calculation is \( O(n) \). \( \Box \)

Based on Theorem 8, we can obtain the following corollary for the case that \( l(r) \) is strictly increasing in \( r \).
The problem \( \text{C} \) is known as a NP-hard problem in strong sense. This implies that the problem \( \text{C} \) is reduced to the 3-PARTITION Problem, which is known as a NP-hard problem in strong sense. This implies that the problem \( \text{C} \) is also NP-hard in strong sense.

Due to the strong NP-hardness of the problem \( \text{C} \), we let the analysis of its solving algorithm as an open problem in this paper. In what follows, we study its special case \( P_2|p_{ij}^r = h(p_j)i(l(r), p_{2j}^r = p_j, IC, ND|C_{max} \) and analyze its solving algorithm. If \( p_{ij}^r = p_{2j}^r = p_j \) and \( IC = [p_j, h(p_j)|0] \), the problem \( P_2|p_{ij}^r = h(p_j)i(l(r), p_{2j}^r = p_j, IC, ND|C_{max} \) is reduced to the problem \( P_2|C_{max} \), which is shown to be a NP-hard problem in Pinedo [20]. Then, we will obtain the following lemma.

**Lemma 5.2.** The problem \( P_2|p_{ij}^r = h(p_j)i(l(r), p_{2j}^r = p_j, IC, ND|C_{max} \) is NP-hard.

By using the standard interchange technology of jobs, the following lemma is easy to prove.

**Lemma 5.3.** The optimal sequencing with respect to the makespan of jobs scheduled on machine 1 is the LPT (longest processing times first) order such that jobs with bigger \( p_j \) are processed earlier.

This lemma indicates a dynamic programming algorithm, which starts from an initial job sequence with \( n \) jobs in the LPT order and can eventually find an optimal
schedule by making either processing on machine 1 or processing on machine 2
decision on jobs one-by-one. For convenience, we let the jobs be indexed in the
LPT order such as \( p_1 \geq p_2 \geq \cdots \geq p_n \). Let \((j, C_1, r, C_2, C_j^\text{max})\) be the state
representative denoting a partial schedule on the first \( j \) jobs, where \( j \leq n \), \( C_i \) is
the completion time of the last processed job on machine \( i \) (\( i = 1, 2 \)), and \( r \) is
the number of jobs processing on machine 1, and \( C_j^\text{max} \) is the makespan. If \( j = n \),
the state \((j, C_1, r, C_2, C_j^\text{max})\) denotes a full schedule and also a feasible solution to
the problem. For \( j < n \), the next unscheduled job \( J_{j+1} \) can be scheduled in the
following ways.

- **Case 1.** Job \( J_{j+1} \) is scheduled on machine 1. State \((j+1, C_1 + f_{j+1}(j+1, r+1) +
\alpha_2 \max(0, f_{j+1}(j+1, r+1) - b_{j+1}), r+1, C_2, C_j^\text{max} + 1)\) is generated, where
\( C_j^\text{max} = C_j^\text{max} + \max(0, C_1 + f_{j+1}(j+1, r+1) + \alpha_2 \max(0, f_{j+1}(j+1, r+1) -
b_{j+1}) - C_j^\text{max}) \).

- **Case 2.** Job \( J_{j+1} \) is scheduled on machine 2. State \((j+1, C_1, r, C_2 +
p_{j+1}, C_j^\text{max} + 1)\) is generated, where \( C_j^\text{max} = C_j^\text{max} + \max(0, C_2 + p_{j+1} - C_j^\text{max}) \).

Before developing an efficient dynamic programming algorithm, the following
easy-to-prove lemma is presented.

**Lemma 5.4.** Consider two states \((j, C_1, r, C_2, C_j^\text{max})\) and \((j', C_1', r', C_2', C_j'^\text{max})\) with
\( 0 < j = j' < n \), \( r = r' \), \( C_j^\text{max} = C_j'^\text{max} \), \( C_1 = C_1' \), and \( C_2 < C_2' \) (or with \( 0 < j = j' <
\), \( r = r' \), \( C_j^\text{max} = C_j'^\text{max} \), \( C_2 = C_2' \), and \( C_1 < C_1' \)), as any later schedules generated
from \((j', C_1', r', C_2', C_j')\) cannot be advantage to the corresponding schedules generated
from \((j, C_1, r, C_2, C_j^\text{max})\), eliminating state \((j', C_1', r', C_2', C_j')\) won’t lead to any
non-optimal solutions for the problem \( P_2 | p_{i,j}' = h(p_j)l(r), p_{2,j}' = p_j, IC, ND | C_{\text{max}} \).

In the following, we propose a dynamic programming algorithm starting from
an empty state \((0, 0, 0, 0, 0) \in S(0)\) (where no job has been processed yet); generates
states \((j, C_1, r, C_2, C_j^\text{max})\) by processing jobs one-by-one; and finds the optimal
schedule by selecting state \((n, C_1, r, C_2, C_j^\text{max})\) with the smallest \( C_j^\text{max} \) among
all states in \( T(n) \). If all the jobs are scheduled on machine 2, the schedule has
the makespan \( C_j^\text{max} = \sum_{j=1}^{n} p_j \). It is obvious that \( \sum_{j=1}^{n} p_j \) is an upper bound of
the optimal solution value. Thus, the following algorithm will delete any state
\((j, C_1, r, C_2, C_j^\text{max})\) with \( C_j^\text{max} > \sum_{j=1}^{n} p_j \).

**Theorem 5.5.** Algorithm 1 finds an optimal solution to the problem \( P_2 | p_{i,j}' =
h(p_j)l(r), p_{2,j}' = p_j, IC, ND | C_{\text{max}} \) in \( O(n \sum_{j=1}^{n} p_j) \) time. This clarifies that the problem
\( P_2 | p_{i,j}' = h(p_j)l(r), p_{2,j}' = p_j, IC, ND | C_{\text{max}} \) is only NP-hard in the ordinary
sense.

**Proof.** The lemmas and discussions in this section directly propose the correctness
of Algorithm 1. For complexity of Algorithm A1, there are in total \( n \) outer loops,
where each state there are at most two options in each loop. As the [Elimination]
procedure eliminates all non-necessary states, the number of candidate states in
\( S(j+1) \) at the beginning of each loop is upper-bounded by \( 2 \sum_{j=1}^{n} p_j \) because that
\( C_j^\text{max} + \max(0, C_1 + f_{j+1}(j+1, r+1) + \alpha_2 \max(0, f_{j+1}(j+1, r+1) - b_{j+1}) - C_j^\text{max}) \leq
\)
Then the bound of the optimal makespan is obtained in the following lemma.

it is shown that

of the problem

P

A

makespan before conducting the FPTAS. In the above analysis of Algorithm

method in Steiner and Zhang [22] and Sahni [21] to convert the FPTAS.

this FPTAS runs

strongly

et al. [6] to convert their dynamic programming algorithm into an FPTAS and

in Sahni [21] and the bound improvement procedure (BIP) developed in Chubanov

cost. Steiner and Zhang [22] used the static partitioning method (SPM) introduced

and FPTASs for the considered single machine scheduling problems with rejection

\[ \sum_{j=1}^{n} p_j \] and \( C_{\max}^j + \max(0, C_2 + p_{j+1} - C_{\max}^j) \) are no more than \( \sum_{j=1}^{n} p_j \), respectively.

Therefore, the overall run time is \( O(2n \sum_{j=1}^{n} p_j) \), which is indeed \( O(n \sum_{j=1}^{n} p_j) \).

In the following, a fully polynomial time approximation scheme (FPTAS) for the

considered problem \( P_2[p_{ij}, h(p_j)l(r), p_{2j} = p_j, IC, ND|C_{\max} \) is developed by

trimming the state space of the Algorithm A1 by fusing states that are ‘close’ to

each other. Woeginger [23] developed a classic general technique, where the fusion

of states is obtained by constraining the jobs in any schedule to finish at times of

the form \( \tau_j = (1 + \varepsilon^j)j \) for \( j \geq 0 \) and \( \varepsilon > 0 \). Engels et al. [7] and Cheng and Sun

[5] used the technique in Woeginger [23] to develop pseudo-polynomial algorithms

and FPTASs for the considered single machine scheduling problems with rejection

cost. Steiner and Zhang [22] used the static partitioning method (SPM) introduced

in Sahni [21] and the bound improvement procedure (BIP) developed in Chubanov et al. [6] to convert their dynamic programming algorithm into an FPTAS and

this FPTAS runs strongly in polynomial time. This section will the similar analysis

method in Steiner and Zhang [22] and Sahni [21] to convert the FPTAS.

Similar with Steiner and Zhang [22], we propose the bound analysis of the

makespan before conducting the FPTAS. In the above analysis of Algorithm A1,

it is shown that \( C_{\max}^* \leq \sum_{j=1}^{n} p_j \), where \( C_{\max}^* \) denotes the optimal solution value

of the problem \( P_2[p_{ij}, h(p_j)l(r), p_{2j} = p_j, IC, ND|C_{\max} \). Because that \( C_{\max} = \max\{C_1, C_2\} \geq (C_1 + C_2)/2 \geq \sum_{j=1}^{n} p_j/2 \), it can be obtained that \( C_{\max}^* \geq (\sum_{j=1}^{n} p_j)/2 \).

Then the bound of the optimal makespan is obtained in the following lemma.

Algorithm A1

| [Initialization] Set the initial state \( S^{(0)} = \{(0,0,0,0,0)\} \), and \( S^{(j)} = \emptyset \) for all \( j = 1, 2, \ldots, n \). |
| [Generation] For \((j, C_1, r, C_2, C_{\max}^j) \in S^{(j)}, j = 0, \ldots, n - 1\), set \( T = \emptyset \) and do:
  | If \( C_{\max}^j = C_{\max}^j + \max(0, C_1 + f_j + 1(j + 1, r + 1) + \alpha_2 max(0, f_j + 1(j + 1, r + 1) - b_{j+1}) \leq \sum_{j=1}^{n} p_j, \)
    | then generate \((j + 1, C_1 + p_{j+1} + 1, r + 1, C_2, C_{\max}^{j+1})\) and set \( T = T \cup \{(j + 1, C_1 + p_{j+1} + 1, r + 1, C_2, C_{\max}^{j+1})\} \). /* Case 1.
  | If \( C_{\max}^{j+1} = C_{\max}^j + \max(0, C_2 + p_{j+1} - C_{\max}^j) \leq \sum_{j=1}^{n} p_j, \)
    | then generate \((j + 1, C_1, r, C_2 + p_{j+1}, C_{\max}^{j+1})\) and set \( T = T \cup \{(j + 1, C_1, r, C_2 + p_{j+1}, C_{\max}^{j+1})\} \). /* Case 2.
| [Elimination] If \( j < n \), for any two states \((j, C_1, r, C_2, C_{\max}^j)\) and \((j', C_1', r', C_{\max}^j')\) with \( 0 < j = j' < n, r = r', \)
  | \( C_{\max}^j = C_{\max}^j, C_1 = C_1' \), and \( C_2 < C_2' \) (or with \( 0 < j = j' < n, r = r', \)
  | \( C_{\max}^j = C_{\max}^j, C_2 = C_2', C_1 < C_1' \)\), eliminate state \((j', C_1', r', C_{\max}^j, C_{\max}^j)\) from \( T \), and then set \( S^{(j+1)} = T \) after all the
  | possible eliminations are conducted. Otherwise, set \( S^{(n)} = T \).
| [Optimization] If \( S^{(n)} \neq \emptyset \), find the state \((n, C_1, r, C_2, C_{\max}^n)\) with the
  | smallest makespan over all states in \( S^{(n)} \). Obtain the optimal
  | solution value and trace back to
  | get the corresponding schedule as the optimal schedule.
In this paper, three multi-machine scheduling problems with interval constrained actual processing times are analyzed. For the first problem of minimizing the total load on machines, four cases are proved to be polynomially solved in the decreasing computational complexities when constraints are added; for the second problem, the objective is to minimize the number of jobs exceeding intervals, this paper shows that five cases of this problem can be also solved in polynomial time. For the makespan minimization problem, this paper proved that the problem is NP-hard in strong sense if the machine number is not restricted, and proposed a dynamic programming algorithm and an FPTAS to solve the special case with two parallel-machine setting, which is showed as NP-hard in ordinary sense.

**Algorithm A2**

- **Initialization** Set $S(0) = \{(0,0,0,0,0)\}$, and $S(j) = \emptyset$ for all $j = 1, 2, \ldots, n$.
- **Partitioning** Partition $[0, U]$ into $\lfloor \frac{2}{\varepsilon} \times n \rfloor$ equal subintervals of size $\frac{2}{n} \times \frac{2}{\varepsilon}$ with the last subinterval possibly smaller.
- **StateGeneration** Do the same as in Algorithm 1.
- **StateElimination** If $j < n$, for the states $(j, C_1, r, C_2, C_{\max})$ with the same $r$ value and $C_{\max} \leq U$ values falling in the same subinterval, keep only the one with the smallest $C_1$ or $C_2$ value, and set $S(j+1) = T$ after all of these operations. Otherwise, set $S(n) = T$.
- **StateSelection** In $S(n)$, find the state $(n, C_1, r, C_2, C_{\max})$ with the smallest $C_{\max}$ value. Set $C_{\max}$ as the solution value and trace back to obtain the corresponding schedule.

**Lemma 5.6.** The optimal makespan of the problem $P_2|p_{ij}^l = h(p_j)l(r), p_{ij}^r = p_j, IC, ND|C_{\max}$ is in the interval $[L, 2L]$, where $L = \sum_{j=1}^{n} p_j/2$.

Implementing SPM in Sahni [21] on Algorithm A1, the above bounds $[L, U]$ will lead to an approximation algorithm, which will find an $(1 + \varepsilon)$-approximation solution to the $P_2|p_{ij}^l = h(p_j)l(r), p_{ij}^r = p_j IC, ND|C_{\max}$ problem in $O(\frac{n^2}{\varepsilon})$ time, where $U = 2L$ and $\varepsilon > 0$. In the following, Algorithm 2 is proposed.

**Remark 1.** In [Partitioning], the interval $[0, 2L]$ is divided into $\lceil \frac{2n}{\varepsilon} \rceil$ subintervals as the length of each subinterval has to be no greater than $\frac{2n}{\varepsilon}$, which is the maximum error allowed to be introduced in each iteration. This gives that the total cumulative error after $n$ iteration won’t be greater than $\varepsilon L$, which achieves the designated approximation ratio $\varepsilon$.

**Theorem 5.7.** For any given $\varepsilon > 0$, the FPTAS A2 finds an $(1 + \varepsilon)$-approximation solution value such as $v \leq (1 + \varepsilon)v^*$ for the problem $P_2|p_{ij}^l = h(p_j)l(r), p_{ij}^r = p_j IC, ND|C_{\max}$. The run time is $O(\frac{n^2}{\varepsilon})$ time.

**Proof.** The correctness of Algorithm A2 directly follows the above discussions and the proof in Sahni [21]. For run time, Sahni [21] showed that $A(L, n, \varepsilon)$ runs in $O(\frac{n^2}{\varepsilon})$ time for the bound interval of objective $[L, nL]$. In Algorithm A2, the bound interval of objective is $[L, 2L]$, its overall run time is reduced to $O(\frac{2n^2}{\varepsilon^2})$, which is indeed $O(\frac{n^2}{\varepsilon})$.

6. **Conclusions.** In this paper, three multi-machine scheduling problems with interval constrained actual processing times are analyzed. For the first problem of minimizing the total load on machines, four cases are proved to be polynomially solved in the decreasing computational complexities when constraints are added; for the second problem, the objective is to minimize the number of jobs exceeding intervals, this paper shows that five cases of this problem can be also solved in polynomial time. For the makespan minimization problem, this paper proved that the problem is NP-hard in strong sense if the machine number is not restricted, and proposed a dynamic programming algorithm and an FPTAS to solve the special case with two parallel-machine setting, which is showed as NP-hard in ordinary sense.
Acknowledgments. The authors would like to express our warmest thanks to the referees for their interest in our work and their valuable comments for improving the paper.

REFERENCES

[1] D. Biskup, Single-machine scheduling with learning considerations, European Journal of Operational Research, 115 (1999), 173–178.
[2] F. Bourgeois and J. C. Lassalle, An extension of the munkres algorithm for the assignment problem to rectangular matrices, Communications of the ACM, 14 (1971), 802–804.
[3] S. Browne and U. Yechiali, Scheduling deteriorating jobs on a single processor, Operations Research, 38 (1990), 495–498.
[4] H. Chen, C. Chu and J. M. Proth, Cyclic scheduling of a hoist with time window constraints, IEEE Transactions on Robotics and Automation, 14 (1998), 144–152.
[5] Y. Cheng and S. Sun, Scheduling linear deteriorating jobs with rejection on a single machine, European Journal of Operational Research, 194 (2009), 18–27.
[6] S. Chubanov, M. Y. Kovalyov and E. Pesch, An fptas for a single-item capacitated economic lot-sizing problem with monotone cost structure, Mathematical Programming, 106 (2006), 453–466.
[7] D. W. Engels, D. R. Karger, S. G. Kolliopoulos, S. Sengupta, R. Uma and J. Wein, Techniques for scheduling with rejection, Journal of Algorithms, 49 (2003), 175–191.
[8] G. Finke, A. Gara-Ali, M. L. Espinouse, V. Jost and J. Moncel, Unified matrix approach to solve production-maintenance problems on a single machine, Omega, 66 (2017), 140–146.
[9] R. L. Graham, Bounds for certain multiprocessing anomalies, Bell System Technical Journal, 45 (1966), 1563–1581.
[10] J. N. D Gupta and S. K. Gupta, Single facility scheduling with nonlinear processing times, Computers & Industrial Engineering, 14 (1988), 387–393.
[11] G. Hardy, J. Littlewood and G. Polya, Inequalities, Cambridge Univ. Press, London, 1965.
[12] M. Ji and T. C. Hsu and D.-L. Yang, Single-machine scheduling with deteriorating jobs and aging effects under an optional maintenance activity consideration, Journal of Combinatorial Optimization, 26 (2013), 437–447.
[13] W. H. Kuo and D. L. Yang, Minimizing the makespan in a single-machine scheduling problem with the cyclic process of an aging effect, Journal of the Operational Research Society, 59 (2008), 416–420.
[14] M. Liu, F. Zheng, C. Chu and J. Zhang, An fptas for uniform machine scheduling to minimize makespan with linear deterioration, Journal of Combinatorial Optimization, 23 (2012), 483–492.
[15] G. Mosheiov, V-shaped policies for scheduling deteriorating jobs, Operations Research, 39 (1991), 979–991.
[16] G. Mosheiov, A note: Multi-machine scheduling with general position-based deterioration to minimize total load, International Journal of Production Economics, 135 (2012), 523–525.
[17] G. Mosheiov and J. B. Sidney, Scheduling with general job-dependent learning curves, European Journal of Operational Research, 147 (2003), 665–670.
[18] J. Munkres, Algorithms for the assignment and transportation problems, Journal of the Society for Industrial and Applied Mathematics, 5 (1957), 32–38.
[19] M. Pinedo, Scheduling, Springer-Verlag, New York, 2012.
[20] S. K. Sahni, Algorithms for scheduling independent tasks, Journal of the ACM, 23 (1976), 116–127.
[21] G. Steiner and R. Zhang, Revised delivery-time quotation in scheduling with tardiness penalties, Operations Research, 59 (2011), 1504–1511.
[22] G. J. Woeginger, When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (fptas)?, INFORMS Journal on Computing, 12 (2000), 57–74.
[23] P. Xue, Y. Zhang and X. Yu, Single-machine scheduling with piece-rate maintenance and interval constrained position-dependent processing times, Applied Mathematics and Computation, 226 (2014), 415–422.
P. Yan, A. Che, X. Cai and X. Tang, Two-phase branch and bound algorithm for robotic cells rescheduling considering limited disturbance, Computers & Operations Research, 50 (2014), 128–140.

P. Yan, A. Che, N. Yang and C. Chu, A tabu search algorithm with solution space partition and repairing procedure for cyclic robotic cell scheduling problem, International Journal of Production Research, 50 (2012), 6403–6418.

P. Yan, C. Chu, N. Yang and A. Che, A branch and bound algorithm for optimal cyclic scheduling in a robotic cell with processing time windows, International Journal of Production Research, 48 (2010), 6461–6480.

Y. Yin, T. Cheng, J. Xu, S. R. Cheng and C. C. Wu, Single-machine scheduling with past-sequence-dependent delivery times and a linear deterioration, Journal of Industrial and Management Optimization, 9 (2013), 323–339.

X. Yu and Y. Zhang, Single machine scheduling with aging effect and upper-bounded actual processing times, Arabian Journal for Science and Engineering, 39 (2014), 1489–1495.

X. Yu, Y. Zhang and K. Huang, Multi-machine scheduling with general position-based deterioration to minimize total load revisited, Information Processing Letters, 114 (2014), 399–404.

X. Zhang, Y. Yin and C. C. Wu, Scheduling with non-decreasing deterioration jobs and variable maintenance activities on a single machine, Engineering Optimization, 49 (2017), 84–97.

Z. Zhou, A. Che and P. Yan, A mixed integer programming approach for multi-cyclic robotic flowshop scheduling with time window constraints, Applied Mathematical Modelling, 36 (2012), 3621–3629.

Received September 2016; revised December 2016.

E-mail address: xyy@nuaa.edu.cn
E-mail address: dlyang@nfu.edu.tw
E-mail address: dqzhou88@163.com
E-mail address: cemzp@nuaa.edu.cn