1. Introduction

A common assumption in classical scheduling theory is that the machines are available at all times. However, there are many situations where machines need to be maintained and become unavailable during certain periods. Cheng et al. [1] study single-machine scheduling with deteriorating job processing times. Lee and Leon [2] investigated single machine scheduling with a rate-modifying activity. Kubzin and Strusevich [3] presented a setting where the maintenance activity was deteriorating; that is, delaying the maintenance activity increased the time required to perform it. Lee and Lin [4] considered single machine scheduling with maintenance and repair rate-modifying activity. Lee and Chen [5] discussed parallel-machine scheduling where each machine must be maintained once during the planning horizon. Cheng et al. [1, 6] considered the unrelated parallel-machine scheduling with aging maintenance activities. Cheng et al. [7] studied a single machine problem of common due-window assignment and scheduling of aging jobs and a maintenance activity simultaneously.

Browne and Yechiali [8] claimed that the time and effort required to put out a fire increase if there was a delay in the start of the fire-fighting effort. Scheduling in this setting is known as scheduling with aging jobs, which was first independently introduced by J. N. D. Gupta and S. K. Gupta [9] and Browne and Yechiali [8]. Since then, models of scheduling with deteriorating or aging jobs have been extensively studied from a variety of perspectives. More recent papers which have considered deterioration include Zhao and Tang [10], Cheng et al. [1], Wang et al. [11], Lai and Lee [12], Cheng et al. [13], Huang and Wang [14], Yin et al. [15], Lee et al. [16], Zhao and Tang [17], Wang and Liu [18], Zhao and Tang [19], Wu and Lee [20], Wang [21], Miao et al. [22], Wu and Lee [23], Yin et al. [24], Oron [25], Sicilia et al. [26], Wang et al. [27], Yin et al. [28], J.-B. Wang and I.-J. Wang [29], Xu et al. [30], Yin et al. [31], Bai et al. [32], and Ji et al. [33].

In modern industrial production, the manufacturing environment has an increasing influence on the processing times of jobs. A growing evidence displays that waiting time or manufacturing environment may have a disadvantageous effect on the total processing time of a job before delivery to the customer. For example, an electronic component may be exposed to certain electromagnetic or radioactive fields while waiting in the machines preprocessing area and regulatory authorities require the component to be treated for an amount of time proportional to the jobs' exposure time.
to these fields. This treatment can be performed after the component has been processed by the machine but before it is delivered to the customer so it can be delivered with a guarantee. Such an extra time for eliminating adverse effects between the main processing and the delivery of a job is viewed as a past-sequence-dependent (psd) delivery time. As usual, we suppose the treatment of the adverse effect for a job does not occupy any machine and has no relation to the schedule of the jobs main processing. In addition, we also assume that the psd delivery time of a job is proportional to the job’s waiting time.

Koulamas and Kyparisis [34] assumed that the psd delivery time of a job is a proportional to the job’s waiting time, that is, the start time of processing. They proved some problems could be polynomially solvable. Liu et al. [35] studied the problem of single-machine scheduling with past-sequence-dependent delivery times, which was introduced in Koulamas and Kyparisis [34]. Liu [36] considered identical parallel-machine scheduling problem with past-sequence-dependent delivery times and learning effect. Shen and Wu [37] introduced the single machine past-sequence-dependent delivery times scheduling with general learning effects. Yin et al. [22] studied a single machine batch scheduling with dependent delivery times and learning effect. Shen and Wu [37] introduced the single machine past-sequence-dependent delivery times scheduling with general learning effects. As in Koulamas and Kyparisis [34], the processing of job \( J_{i,j} \) must be followed by the psd delivery time \( q_{i,r} \), which can be computed as

\[
q_{i,1} = 0, \quad q_{i,r} = \gamma W_{i,r} = \gamma \sum_{l=1}^{r-1} P_{i,l}, \quad r = 2, 3, \ldots, n.
\]

\( \gamma \geq 0 \) is a normalizing constant and \( W_{i,r} \) denotes the waiting time of job \( J_{i,r} \). In addition, it is supposed that the postprocessing operation of any job \( J_{i,j} \) modeled by its delivery time \( q_{i,j} \) is performed off-line; consequently, it is not affected by the availability of the machine and it can commence immediately upon completion of the main operation resulting in

\[
C_{i,j} = W_{i,j} + P_{i,j} + q_{i,j} = (1 + \gamma) \sum_{l=1}^{j-1} P_{i,l} + P_{i,j} \quad \text{if} \quad j \leq k_i,
\]

\[
C_{i,j} = (1 + \gamma) \left( \mu + (1 + \sigma) \sum_{l=k_i+1}^{k_j} P_{i,l} \right) + P_{i,j} \quad \text{if} \quad j > k_i,
\]

(3)

where \( C_{i,j} \) denotes the completion time of job \( J_{i,j} \).

For convenience, we denote the psd delivery times given in (2) by \( q_{psd} \). Let \( TADC_i \) denote the total absolute deviation of jobs’ completion times on machine \( M_i \); that is,

\[
\sum_j TADC_i = \sum_{j=1}^{n_i} \sum_{l=1}^{k_j} |C_{i,l} - C_{i,k_j}|. \quad \text{Let} \quad L_i \quad \text{indicate the load of machine} \quad M_i; \quad \text{that is,} \quad L_i = \max[C_{i,j}], \quad r = 1, 2, \ldots, n_i. \quad \text{We, respectively, consider the minimization of the following objective functions: the sum of the total absolute deviation of jobs’ completion times on each machine} \quad \sum_{i=1}^{m} TADC_i, \quad \text{the total loads on all machines} \quad \sum_{i=1}^{m} L_i, \quad \text{and the total completion time} \quad \sum_{i=1}^{m} \sum_{j=1}^{n_i} C_{i,j}. \quad \text{Using the three-field notation introduced}
by Graham et al. for scheduling problems, we denote our problems as
\[ P_{m} | q_{\text{psd}}, P_{[j]} = a_{[j]} + bt, ma | f, \]
\[ f \in \{ \sum \sum C_{ij}, \sum L_{i}, \sum \text{TADC}_{i} \}. \]

Before presenting the main results, we first present several Lemmas and some notations that will be used in the proofs in sequel. If the number of jobs \( n \) and the position of the job preceding the maintenance operation \( k \) on machine \( M \) are known in advance, the actual processing times and the completion times of jobs on machine \( M \) are as follows:

\[ P_{[1]} = a_{[1]} \]
\[ C_{[1]} = P_{[1]} = a_{[1]} \]
\[ P_{[2]} = a_{[2]} + bP_{[1]} = a_{[2]} + ba_{[1]} \]
\[ C_{[2]} = a_{[2]} + (1 + y + b)a_{[1]} \]
\[ \vdots \]
\[ P_{[k]} = a_{[k]} + ba_{[k-1]} + b(1 + b)a_{[k-2]} + \cdots + b(1 + b)^{k-2}a_{[1]} \]
\[ C_{[k]} = a_{[k]} + (1 + y + b)a_{[k-1]} + \cdots + (1 + y + b)(1 + b)^{k-2}a_{[1]} \]
\[ f(t) = \mu + \sigma \left( P_{[1]} + P_{[2]} + \cdots + P_{[k]} \right) \]
\[ = \mu + \sigma \sum_{j=1}^{k} (1 + b)^{k-j}a_{[j]} \]
\[ P_{[k+1]} = a_{[k+1]} \]
\[ C_{[k+1]} = (1 + y)\mu + (1 + y)(1 + \sigma)\sum_{j=1}^{k} (1 + b)^{k-j}a_{[j]} + a_{[k+1]} \]
\[ \vdots \]
\[ P_{[n]} = a_{[n]} + ba_{[n-1]} + \cdots + b(1 + b)^{n-2}a_{[n+1]} \]
\[ C_{[n]} = (1 + y)\mu + (1 + y)(1 + \sigma)\sum_{j=1}^{k} (1 + b)^{k-j}a_{[j]} + \sum_{j=k+1}^{n-1} (1 + y + b)(1 + b)^{n-j-1}a_{[j]} + a_{[n]} \]

Let \( P(n, m, k) = (n_{1}, n_{2}, \ldots, n_{m}; k_{1}, k_{2}, \ldots, k_{m}) \) denote the allocation vector. We derive a lemma to bound \( P(n, m, k) \).

**Lemma 1.** The number of \( P(n, m, k) \) vector is bounded from above by \((n + 1)^{2m-1}/m!\).

Proof. For machine \( M_{1} \), \( n_{1} \) is chosen from \([0, 1, \ldots, n]\) and there are \( n + 1 \) options for \( n_{1} \). When \( n_{1} \) is determined, \( k_{1} \) has at most \( n + 1 \) options. After \( n_{1} \) is determined, \( n_{2} \) is chosen from \([0, 1, \ldots, n - n_{1}] \) at most \( n + 1 \) times. Subsequently, \( k_{2} \) has at most \( n + 1 \) options, too. For machines \( M_{3}, \ldots, M_{m-1} \), we similarly select \( n_{3}, \ldots, n_{m-1} \) and \( k_{3}, \ldots, k_{m-1} \). Note that, for the last machine \( M_{m} \), \( n_{m} \) is determined after choosing \( n_{1}, \ldots, n_{m-1} \), because \( \sum_{i=1}^{m} n_{i} = n \), and \( k_{m} \) has at most \( n + 1 \) options. Considering that \( m \) machines are identical, there are \((n + 1)^{2m-1}/m!\) possibilities for \( P(n, m, k) \) vector. This completes the proof.

**Lemma 2** (see \([40]\)). Let there be two sequences of numbers \( x_{i}, y_{i} \). The sum \( \sum_{i} x_{i}y_{i} \) of products of the corresponding elements is the least (largest) if the sequences are monotonic in the opposite (same) sense.

3. Parallel-Machine Scheduling Problems

3.1. The Total Completion Time. In this section, we consider the problem of minimizing the total completion time; that is, \( P_{m} | q_{\text{psd}}, P_{[j]} = a_{[j]} + bt, ma | \sum \sum C_{ij} \). For machine \( M_{1} \), from the above analyse, we derive the total completion time on this machine:

\[ \sum_{j=1}^{n_{i}} C_{ij} = (n_{i} - k_{i})(1 + y)\mu + \sum_{j=1}^{k_{i}} \left( 1 + (1 + y + b)(1 + b)^{k_{i}-j} - 1 \right) \frac{a_{[j]}}{b} + \sum_{j=k_{i}+1}^{n_{i}} \left( 1 + (1 + y + b)(1 + b)^{n_{i}-j-1} - 1 \right) a_{[j]}. \]

Therefore, the total completion time is

\[ \sum_{i=1}^{m} \sum_{j=1}^{n_{i}} C_{ij} = \sum_{i=1}^{m} (n_{i} - k_{i})(1 + y)\mu + \sum_{i=1}^{m} \sum_{j=1}^{k_{i}} \left( 1 + (1 + y + b)(1 + b)^{k_{i}-j} - 1 \right) \frac{a_{[j]}}{b} + \sum_{i=1}^{m} \sum_{j=k_{i}+1}^{n_{i}} \left( 1 + (1 + y + b)(1 + b)^{n_{i}-j-1} - 1 \right) a_{[j]}. \]
where

\[
\begin{align*}
   w_{ij} &= \left\{ \begin{array}{ll}
   1 + (1 + y + b) \left( \frac{1}{b} + (n_i - k_j) (1 + y) \right) & \\
   \times (1 + \sigma) \left( 1 + b^{k_{r-j}} \right) & \\
   i = 1, 2, \ldots, m, & j = 1, 2, \ldots, k_i, \\
   \end{array} \right. \\
   &\quad + (n_i - 1) a_i[j], \\
   &\quad + \sum_{i=1}^{m} \sum_{j=1}^{k_{r-j}} (1 + y + b) \left( 1 + b^{k_{r-j}} \right) a_i[j], \\
   &\quad + \sum_{i=1}^{m} a_i[n_i], \\
   \end{align*}
\]

(8)

When \( n_i \) and \( k_i \) are given, \( \sum_{j=1}^{m} \sum_{i=1}^{k_i} w_{ij} a_{i[j]} \) can be viewed as the scalar product of the \( w_{ij} \) vector and \( a_{i[j]} \) vector. Therefore, on machine \( M_i \), if the number \( n_i \) of jobs and the maintenance position \( k_i \) are known, by Lemma 2, all jobs are sorted in nondecreasing order of their normal processing times first. Then the largest job is assigned to the position with the smallest value of \( w_{ij} \), the second largest job to the position with the second smallest value of \( w_{ij} \), and so on. The time complexity of sorting algorithm on machine \( M_i \) is \( O(n_i \log n_i) \). Together with Lemma 1, we derive the following theorem.

**Theorem 3.** The problem \( P_m | q_{p,d}, p_{i[j]} = a_{i[j]} + bt, ma | \sum C_{ij} \) can be solved in \( O(n^{2m-1} n \log n) \) time.

### 3.2. The Sum of Load on All Machines

In this section, we investigate the problem of minimizing the sum of load on all machines; that is, \( P_m | q_{p,d}, p_{i[j]} = a_{i[j]} + bt, ma | \sum L_i \). For machine \( M_i \), from the above analysis, we calculate the sum of load on this machine:

\[
L_i = (1 + y) \mu \\
+ (1 + y) (1 + \sigma) \sum_{j=1}^{k_i} (1 + b)^{k_{r-j}} a_{i[j]} \\
+ \sum_{j=k_i+1}^{n_i} (1 + y + b) (1 + b)^{n_i-j-1} a_{i[j]} + a_{i[n_i]},
\]

(9)

Hence, the sum of load on all machines is

\[
\begin{align*}
   \sum_{i=1}^{m} L_i &= m (1 + y) \mu \\
   &\quad + \sum_{i=1}^{m} \sum_{j=1}^{k_i} (1 + y) (1 + \sigma) (1 + b)^{k_{r-j}} a_{i[j]} \\
   &\quad + \sum_{i=1}^{m} \sum_{j=k_i+1}^{n_i} (1 + y + b) (1 + b)^{n_i-j-1} a_{i[j]} \\
   &\quad + \sum_{i=1}^{m} a_{i[n_i]} = m (1 + y) \mu \\
   &\quad + \sum_{i=1}^{m} \sum_{j=1}^{k_i} \sum_{h=j}^{k_i} (1 + y + b) (1 + b)^{k_{r-j}} a_{i[j]} \\
   &\quad + \sum_{i=1}^{m} a_{i[n_i]}.
\end{align*}
\]

(10)

Since \( m(1 + y) \mu \) is a constant, if \( n_i \) and \( k_i \) are given, \( \sum_{i=1}^{m} \sum_{j=1}^{k_i} w_{ij} a_{i[j]} \) can be viewed as the scalar product of the \( w_{ij} \) vector and \( a_{i[j]} \) vector. Similar to the above analysis, on machine \( M_i \), the time complexity of sorting algorithm is \( O(n_i \log n_i) \). Together with Lemma 1, we conclude the following theorem.

**Theorem 4.** The problem \( P_m | q_{p,d}, p_{i[j]} = a_{i[j]} + bt, ma | \sum L_i \) can be solved in \( O(n^{2m-1} n \log n) \) time.

### 3.3. The Sum of TADC on All Machines

In this section, we present the problem of minimizing the sum of TADC on all machines; that is, \( P_m | q_{p,d}, p_{i[j]} = a_{i[j]} + bt, ma | \sum TADC_i \). For machine \( M_i \), from the above analysis, we express the sum of TADC on this machine:

\[
\begin{align*}
   \text{TADC}_i &= \sum_{j=k_i+1}^{n_i} (2j - 1 - n_i) (1 + y) \mu \\
   &\quad + \sum_{j=1}^{k_i} \left( 2j - 1 - n_i \right) \left( 2h - 1 - n_i \right) (1 + y + b) \\
   &\quad \times (1 + b)^{k_{r-j}} \\
   &\quad + \sum_{h=k_i+1}^{n_i} (2h - 1 - n_i) (1 + y) \\
   &\quad \times (1 + \sigma) (1 + b)^{k_{r-j}} a_{i[j]} \\
   &\quad + \sum_{j=1}^{k_i} \sum_{h=j}^{k_i} (2j - 1 - n_i) (1 + y + b) \\
   &\quad \times (1 + b)^{k_{r-j}} a_{i[j]} \\
   &\quad + (n_i - 1) a_{i[n_i]},
\end{align*}
\]

(11)
Thus, the sum of TADC on all machines is
\[ \sum_{i=1}^{m} TADC_i = \sum_{i=1}^{m} \sum_{j=k_i+1}^{n_i} (2j-1-n_i)(1+\gamma)\mu \]
\[ + \sum_{i=1}^{m} \sum_{j=k_i+1}^{n_i} (2j-1-n_i) \]
\[ + \sum_{h=j+1}^{k_i} (2h-1-n_i)(1+\gamma+b)(1+b)^{h-j-1} \]
\[ + \sum_{h=k_i+1}^{n_i} (2h-1-n_i)(1+\gamma)(1+\sigma) \]
\[ \times (1+b)^{h-j} a_{i[j]} \]
\[ + \sum_{i=1}^{m} \sum_{j=k_i+1}^{n_i} (2j-1-n_i) \]
\[ + \sum_{h=j+1}^{n_i} (2h-1-n_i) \]
\[ + \sum_{h=k_i+1}^{n_i} (2h-1-n_i)(1+\gamma+b) \]
\[ \times (1+b)^{h-j} a_{i[j]} \]
\[ + \sum_{i=1}^{m} (n_i-1) a_{i[n_i]} \]
\[ = \sum_{i=1}^{m} \sum_{j=k_i+1}^{n_i} (2j-1-n_i)(1+\gamma)\mu \]
\[ + \sum_{i=1}^{m} \sum_{j=1}^{n_i} w_{ij} a_{i[j]}, \] 
(13)

where
\[ w_{ij} = \begin{cases} 
(2j-1-n_j) & + \sum_{h=j+1}^{k_i} (2h-1-n_i)(1+\gamma+b)(1+b)^{h-j-1} \\
+ \sum_{h=k_i+1}^{n_i} (2h-1-n_i)(1+\gamma)(1+\sigma)(1+b)^{h-j}, & i = 1, 2, \ldots, m, \quad j = 1, 2, \ldots, n_i, \\
(2j-1-n_j) & + \sum_{h=j+1}^{k_i} (2h-1-n_i)(1+\gamma+b)(1+b)^{h-j-1}, \\
+ \sum_{h=k_i+1}^{n_i} (2h-1-n_i)(1+\gamma+b)(1+b)^{h-j-1}, & i = 1, 2, \ldots, m, \quad j = k_j + 1, k_j + 2, \ldots, n_j - 1, \\
\sum_{i=1}^{m} (n_i-1), & i = 1, 2, \ldots, m, \quad j = n_i.
\end{cases} \] 
(14)

Since \( \sum_{i=1}^{m} \sum_{j=1}^{n} (2j-1-n_i)(1+\gamma)\mu \) is a constant, when \( n_i \) and \( k_i \) are given, \( \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} a_{i[j]} \) can be viewed as the scalar product of the \( w_{ij} \) vector and \( a_{i[j]} \) vector. Similar to the above analysis, the time complexity of scheduling algorithm on machine \( M_i \) is \( O(n_i \log n_i) \). Together with Lemma 1, we obtain the following theorem.

**Theorem 5.** The problem \( P_{m|q_{psd},|p_{ij}} = a_{i[j]} + b, m a| \sum TADC_i \) can be solved in \( O(n^{2m-1} n \log n) \) time.

4. Conclusions

In this paper we study parallel-machine scheduling problems with past-sequence-dependent (psd) delivery times and aging maintenance. The delivery time of a job is proportional to its waiting time. Each machine has an aging maintenance activity. Polynomially solvable problems have been explored for parallel-machine scheduling. The research extends the existed models of parallel machine scheduling with delivery times. For future research, it will be worth extending the problem to multiple maintenances or flow-shop environment.

**Conflict of Interests**

The authors declare that there is no conflict of interests regarding the publication of this paper.

**Acknowledgments**

The work was partly supported by the National Natural Science Foundation of China (71071113), Ph.D. Programs Foundation of the Ministry of Education of China (20100072110011), Shanghai Philosophical and Social Science Foundation of the Ministry of Education of China (71071113), Ph.D. Program of Central Universities. The authors would like to thank the anonymous referees for their constructive comments and suggestion.

**References**


Submit your manuscripts at http://www.hindawi.com