


## Research Article

# Two Parallel-Machine Scheduling Problems with Function Constraint

Chia-Lun Hsu <sup>1,2</sup> and Jan-Ray Liao<sup>1</sup>

<sup>1</sup>Department of Electrical Engineering, National Chung-Hsing University, Tai-Chung 402, Taiwan

<sup>2</sup>Aeronautical Systems Research Division, Chung Shan Institute of Science and Technology, Tai-Chung 402, Taiwan

Correspondence should be addressed to Chia-Lun Hsu; [d9864002@mail.nchu.edu.tw](mailto:d9864002@mail.nchu.edu.tw)

Received 5 April 2020; Accepted 29 April 2020; Published 18 May 2020

Guest Editor: Yunqiang Yin

Copyright © 2020 Chia-Lun Hsu and Jan-Ray Liao. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The objective of this paper is to minimize both the makespan and the total completion time. Since parallel-machine scheduling which contains the function constraint problem has been a new issue, this paper explored two parallel-machine scheduling problems with function constraint, which refers to the situation that the two machines have a same function but one of the machines has another. We pointed out that the function constraint occurs not only in the manufacturing system but also in the service system. For the makespan problem, we demonstrated that it is NP-hard in the ordinary sense. In addition, we presented a polynomial time heuristic for this problem and have proved its worst-case ratio is not greater than  $5/4$ . Furthermore, we simulated the performance of the algorithm through computational testing. The overall mean percent error of the heuristic is 0.0565%. The results revealed that the proposed algorithm is quite efficient. For the total completion time problem, we have proved that it can be solved in  $O(n^4)$  time.

## 1. Introduction

The scheduling problem studied in this paper was motivated by the manufacturing of the metal processing industry. In the traditional manufacturing, a lathe machine and a milling machine have different functions. Generally speaking, the lathe machine is a tool that rotates a workpiece about an axis of rotation to perform various operations such as cutting, deformation, knurling sanding, and turning. The milling machine is used to cut the plane, while the shape of the forming surface, the spiral groove, or the tooth shape of the gear is milled with a special-shaped milling cutter. During milling, the workpiece is mounted on a table or an indexing head attachment, and the milling cutter performs a cutting motion, supplemented by a table for feeding motion. And now, a 5-axis machining center for milling and turning has all the functions of lathe machine and milling machine. As the company's performance grows rapidly, companies must purchase machines to meet customer needs. While newly purchased machines tend to have more function than older

machines, functional alternatives occur between machines. We named this phenomenon as function constraint (e.g., lathe machine versus 5-axis machining centers for milling and turning). The function constraint occurs not only in the manufacturing system but also in the service system. In the service system, the parallel-machine is composed by the employees. The senior staff, the junior employees, and the new employees all have the ability to limit and replace.

Parallel machine in a production environment can be divided into three categories according to the nature of the machine: identical, uniform, and unrelated parallel machines. For decades, the parallel-machine production scheduling problem has been extensively studied under various classical scheduling performance measurement criteria. The makespan and total completion time are the two best important performance measurement criteria. The makespan is also called the maximum completion time or the completion time of the last workpiece on the last machine. The makespan is usually used to measure the utilization of machinery and equipment. If one shortens the

makespan and the machine utilization and production efficiency is improved, more flexible time can be reserved to prevent the sudden occurrence of the production line. The completion time refers to the time spent in the system from the time of the workpiece that arrives at the site to its completion, and the total completion time is the sum of the completion time of all the workpieces. Therefore, the total completion time is expected to be minimized. That is, the in-process inventory of the factory is expected to be minimized to reduce the cost of inventory.

Chung et al. [1] explored a two identical parallel-machine scheduling problem with molds constraints. Their objective is to minimize the makespan. For the problem is NP-hard, they give three heuristics and analyze each heuristic has a worst-case performance ratio of  $3/2$ . Computational results revealed that heuristics are efficient even for the large-sized problem. Xu and Yang [2] studied a two parallel-machine scheduling with a periodic availability constraint. Their objective is to minimize the makespan. Because the problem is NP-hard, they present a mathematical programming model to solve it and, next, compare the performance of the longest processing time first (LPT) algorithm with the list scheduling (LS) via computational experiments. Most of the results showed that the LPT is better than the LS. Xu et al. [3] examined a two parallel machine scheduling problem to minimize the makespan. The problem is known as NP-hard. They applied the branch-and-bound method to solve the small-sized problem and presented a Tabu search algorithm for the large-sized problem. Ji and Cheng [4] presented a fully polynomial-time approximation scheme for parallel-machine scheduling under a grade of service provision to minimize makespan consideration. Lee et al. [5] addressed a makespan minimization scheduling problem on identical parallel-machine. They applied the simulated annealing method, and several heuristic algorithms have been proposed to tackle the problem. Computational results demonstrated that the simulated annealing method is efficient and better than the existing methods. Yin et al. [6] explored parallel-machine scheduling of deteriorating jobs with potential machine disruptions. The authors examined two cases of machine disruption (i.e., performing maintenance immediately on the disrupted machine when a disruption occurred and not performing machine maintenance). The nature of the jobs has two types: nonresumable and resumable. They determined the computational complexity status of various cases of the total completion time minimization problem and provided pseudopolynomial-time solution algorithms and fully polynomial-time approximation schemes for them.

Zhao et al. [7] explored a two parallel-machine scheduling problem where one machine is not available in a specified time period. The unavailable time period is fixed and known in advance. They proposed a fully polynomial-time approximation scheme for the total weighted completion time minimization problem and generalized the results to the case with  $m$  parallel machine. Kuo and Yang [8] studied parallel-machine scheduling with time-dependent processing time. For the total completion time

and the total load problems, they showed that the two problems are polynomially solvable. Gerstl and Mosheiov [9] proposed a general position-dependent processing time model. They considered scheduling problems which combine the option of job rejection and the model on parallel machine. Their objectives are total flow-time and total load. They proved that both problems can be solved in polynomial time in the number of jobs. Huang and Wang [10] stressed parallel-machine scheduling problems with deteriorating jobs. They showed that the total absolute differences in completion time and the total absolute differences in waiting time minimization problems can be solved in polynomial time.

Wang and Wang [11] studied a three-machine permutation flow shop scheduling problem with time-dependent processing times. The objective is to find a sequence that minimizes the makespan. Several dominance properties and a lower bound are derived to speed up the elimination process of a branch-and-bound algorithm. Moreover, two heuristic algorithms are proposed to overcome the inefficiency of the branch-and-bound algorithm. Computational results show that the heuristic algorithm M-NEH performs effectively and efficiently.

For more information, the reader may refer to the concise surveys on this topic by Cheng and Sin [12], Mokotoff [13], Pfund et al. [14], Kravchenko and Werner [15], Kaabi and Harrath [16], and Wang and Li [17].

Parallel-machine scheduling combines the function constraint problem is a new issue. Therefore, this paper explored two parallel-machine scheduling problems with function constraint. The objectives are to minimize the makespan and the total completion time.

The rest of the paper is organized as follows. In Section 2, we introduced the notation and formally formulated our problems. In Section 3, we demonstrated the computational complexity status and presented a heuristic algorithm, worst-case ratio, and computational experiments for the makespan minimization problem. In Section 4, we formulated the total completion time minimization problem as an assignment. In the last section, we concluded the paper and suggested issues for the future research.

## 2. Problem Formulation

There are  $n$  jobs in the set  $N = (J_1, J_2, \dots, J_{n_1}, J_{n_1+1}, \dots, J_n)$ . Assume set  $N$  consists of two classes of job that are the first class jobs and the second class jobs. Let  $n_1$  and  $n_2$  denote the number of jobs of the first class and the second class, respectively. Let  $S_1$  and  $S_2$  denote the set of the jobs of the first class and the second class, respectively. That is,  $N = S_1 \cup S_2$ , where  $S_1 = (J_1, J_2, \dots, J_{n_1})$  and  $S_2 = (J_{n_1+1}, \dots, J_n)$ . Let  $p_j$  denote the processing time of  $J_j$  ( $j = 1, 2, \dots, n$ ). Assume there are two parallel machines  $M_i$  ( $i = 1, 2$ ), where  $M_1$  can process the first class jobs only and  $M_2$  can process both of the first class jobs and the second class jobs. Some common assumptions are as follows: (1) all jobs are non-preemptive and available for processing at time zero; (2) each machine can handle at most one job at a time and cannot stand idle until the last job assigned to it has finished processing; and

(3) each job can be processed on at most one machine at a time. The objectives are to minimize the makespan and the total completion time. We used “fc” to denote the function constraint relations between the two machines. Following the common scheduling notation, the problems can be described as  $P2|fc|C_{\max}$  and  $P2|fc|TC$ , respectively.

### 3. Minimization of the Makespan

In this section, we confirmed the complexity and presented the worst-case bound for  $P2|fc|C_{\max}$ . First, we excluded the case of the sum of processing time in the second class jobs is larger than the sum of processing time in the first class jobs because the first class jobs processed on  $M_1$  and the second class jobs processed on  $M_2$  are optimal for the abovementioned case. Second, we analyze the complexity of the problem  $P2|fc|C_{\max}$ . Third, we presented a heuristic approach and analyze the worst-case bound of the algorithm. Finally, a computational experiment was conducted to evaluate the performance of the proposed algorithm.

**3.1. The Heuristic and Worst-Case Bound.** The special case of all the jobs belongs to the first class,  $n_1 = n$ , and then the problem  $P2|fc|C_{\max}$  becomes the classical problem  $P2||C_{\max}$ . It is known in advance that  $P2||C_{\max}$  is NP-hard in the ordinary sense [18]. Therefore, the following theorem holds.

**Theorem 1.** *Scheduling problem  $P2|fc|C_{\max}$  is NP-hard in the ordinary sense.*

A simple heuristic approach is described in Algorithm 1.

A simple sort can be done in  $O(n \log n)$  (e.g., heap sorting). Hence, the complexity of Step 1 is  $O(n \log n)$ . The complexity of Step 2 is  $O(n)$ . Overall, the complexity of the algorithm is  $O(n \log n)$ .

A straightforward result is given as follows.

**Lemma 1.** *For the problem  $P2|fc|C_{\max}$ , the following holds:*

- (1)  $C_{\max}^* \geq \sum_{j=n_1+1}^n p_j$ , where  $C_{\max}^*$  denotes the makespan of the optimal schedule
- (2)  $C_{\max}^* \geq (\sum_{j=1}^{n_1} p_j + \sum_{j=n_1+1}^n p_j)/2 = LB$ , where  $LB$  denotes the lower bound of the problem

**Theorem 2.** *For the problem  $P2|fc|C_{\max}$ ,  $C_{\max}(LPT) \leq 5C_{\max}^*/4$ .*

*Proof.* Assume  $J_k$  is the last processed job based on the LPT approach. If  $J_k \in S_2$ , then  $C_{\max}(LPT) = C_{\max}^*$ . Therefore, the following proof processes are under the assumption of  $J_k \in S_1$ .

If  $k = 1$ , then  $C_{\max}(LPT) = C_{\max}^*$ .

For the case of  $k \geq 2$ , assume  $s_k$  denotes the starting time of  $J_k$ . Since there is no idle time on the machines prior to the time  $s_k$ ,

$$\begin{aligned} s_k &\leq \frac{(\sum_{j=1}^{k-1} p_j + \sum_{j=n_1+1}^n p_j)}{2} \\ &= \frac{(\sum_{j=1}^k p_j + (\sum_{j=n_1+1}^n p_j) - p_k)}{2} \\ &\leq \frac{(\sum_{j=1}^{n_1} p_j + \sum_{j=n_1+1}^n p_j - p_k)}{2} \\ &\leq C_{\max}^* - \frac{p_k}{2}. \end{aligned} \tag{1}$$

Consider the following two cases:  $p_k \leq C_{\max}^*/2$  and  $p_k > C_{\max}^*/2$ .

When  $p_k \leq C_{\max}^*/2$ , we have

$$\begin{aligned} C_{\max}(LPT) &= s_k + p_k \\ &\leq C_{\max}^* - \frac{p_k}{2} + p_k \\ &= C_{\max}^* + \frac{p_k}{2} \\ &\leq C_{\max}^* + \frac{C_{\max}^*}{4} \\ &= 5 \frac{C_{\max}^*}{4}. \end{aligned} \tag{2}$$

When  $p_k > C_{\max}^*/2$ , based on optimal consideration, we have  $\sum_{j=n_1+1}^n p_j \leq C_{\max}^*/2$  and each machine cannot arrange two jobs that belong to  $S_1$  and whose job's index is less or equal to  $k$ . Therefore,  $J_k$  is the only job that belongs to  $S_1$  and being processed on  $M_2$ . That is,  $C_{\max}(LPT) = p_k + \sum_{j=n_1}^n p_j$ . Within the optimal sequence,  $J_k$  is the only job that belongs to  $S_1$  and can be processed on  $M_2$ . We have  $C_{\max}^* \geq C_{\max}(LPT)$ . Hence,  $C_{\max}^* = C_{\max}(LPT)$ .  $\square$

**3.2. The Computational Experiments.** Although the worst case above seems quite good from the theoretical aspects, 25% errors cannot be accepted from the application dimensions. Therefore, some computational experiments were conducted to evaluate the performance of the LPT approach. The LPT approach was coded in Visual BASIC 6.0 and implemented on a personal computer with Intel core i7 16G CPU. Some test problems for each environment were randomly generated, the details of which are as follows:

- (1)  $n$  is equal to 50, 100, 150, 200, 250, 300, 500, and 1000
- (2)  $n_1$  is uniformly distributed over  $[1, n]$ .  $n_2 = n - n_1$
- (3)  $p_i$  is uniformly distributed over  $[1, 50]$ ,  $[1, 100]$ ,  $[1, 200]$ ,  $[1, 500]$ , and  $[1, 1000]$

A ratio of  $((C_{\max}(LPT) - LB) \times 100)/LB$  is used as an index to evaluate the performance of the LPT approach, where  $C_{\max}(LPT)$  is the solution found by the LPT approach and  $LB$

Step 1. Arrange the first class jobs in nonincreasing order, without loss of generality, and assume the LPT sequence remains  $J_1, J_2, \dots, J_{n_1}$ .  
 Step 2. Assign  $J_{n_1+1}, J_{n_1+2}, \dots, J_n$  to the  $M_2$ . After that, whenever a machine is freed, the longest job among those not yet processed in the first class is put on the machine.

ALGORITHM 1: The longest processing time first (LPT) approach.

TABLE 1: Computational results of the LPT approach with five different processing time distribution periods.

$n$	$p(i)$					Average
	$U$ (1, 50)	$U$ (1, 100)	$U$ (1, 200)	$U$ (1, 500)	$U$ (1, 1000)	
50	0.0971	0.1884	0.2817	0.3714	0.4576	0.2792
100	0.0270	0.0492	0.0715	0.0939	0.1167	0.0716
150	0.0156	0.0264	0.0372	0.0483	0.0580	0.0371
200	0.0105	0.0174	0.0236	0.0295	0.0350	0.0232
250	0.0081	0.0132	0.0175	0.0211	0.0247	0.0169
300	0.0066	0.0107	0.0137	0.0163	0.0189	0.0132
500	0.0040	0.0059	0.0071	0.0080	0.0089	0.0068
1000	0.0022	0.0033	0.0038	0.0041	0.0043	0.0035
Average	0.0214	0.0397	0.0570	0.0741	0.0905	0.0565

is the solution found by lower bound. The ratio is computed for 300 test problems in each problem size; hence, 12000 ( $300 \times 8 \times 5$ ) test problems are generated for the proposed problem  $P2|fc|C_{\max}$ . The computational results are shown in Table 1. The results revealed that the mean percentage errors of the LPT approach are 0.0214, 0.0397, 0.0560, 0.0741, and 0.0905 with five different processing time distribution periods, respectively. From Table 1, we found that the larger the problem the smaller the error percentages and the wider the processing time range the larger the error percentages. The overall mean percent error of the LPT approach is 0.0565%.

#### 4. Minimization of the Total Completion Time

In this section, we showed that the problem  $P2|fc|TC$  can be solved in  $O(n^4)$  time. Let  $(k_1, k_2)$  denotes the vector of the number of jobs allocated to  $M_1$  and  $M_2$ , where  $1 \leq k_1 \leq n_1$  and  $n_2 \leq k_2 \leq n - 1$ . We excluded the case of  $(k_1, k_2) = (0, n)$  because it becomes a single machine scheduling problem and the SPT (shortest processing time first) is optimal. Let  $J_{[ij]}$  and  $p_{[ij]}$  denote a job scheduled in the  $j$ th position on a machine  $i$  and its processing time, respectively. For the convenience, we rename the job sequences processing on  $M_1$  and  $M_2$  as  $J_{[11]}, J_{[12]}, \dots, J_{[1k_1]}$  and  $J_{[21]}, J_{[22]}, \dots, J_{[2k_2]}$ , respectively. Then, the total completion time can be calculated as follows:

$$TC = \sum_{j=1}^{k_1} C_j = \sum_{j=1}^{k_1} (k_1 - j + 1)p_{[1j]} + \sum_{j=1}^{k_2} (k_2 - j + 1)p_{[2j]}. \quad (3)$$

We rewrite equation (3) as

$$TC = \sum_{j=1}^n C_j = \sum_{j=1}^n w_j p_{[j]}, \quad (4)$$

where

$$w_j = \begin{cases} k_1 - j + 1, & 1 \leq j \leq k_1, \\ n - j + 1, & k_1 + 1 \leq j \leq n, \end{cases} \quad (5)$$

$$p_{[j]} = \begin{cases} p_{[1j]}, & 1 \leq j \leq k_1, \\ p_{[2j-k_1]}, & k_1 + 1 \leq j \leq n. \end{cases}$$

Assume the position of job that is processing on  $M_1$  and  $M_2$  is  $1, \dots, k_1$  and  $k_1 + 1, k_1 + 2, k_1, k_2(n)$ , respectively. Obviously, the second class of jobs cannot be scheduled at the previous position of  $k_1$ . Then, we gave the elements of assignment vector as follows:

$$a_{ij} = \begin{cases} w_j p_i, & 1 \leq i \leq n_1, 1 \leq j \leq n, \\ \infty, & n_1 + 1 \leq i \leq n, 1 \leq j \leq k_1, \\ w_j p_i, & n_1 + 1 \leq i \leq n, k_1 + 1 \leq j \leq n, \end{cases} \quad (6)$$

where

$$w_j = \begin{cases} k_1 - j + 1, & 1 \leq j \leq k_1 \\ n - j + 1, & k_1 + 1 \leq j \leq n. \end{cases} \quad (7)$$

Constraint of the second line at  $a_{ij}$  ensures that the second class jobs are not scheduled before the position of  $k_1$  under minimization consideration. For a given vector  $(k_1, k_2)$ ,  $P2|fc|TC$  can be formulated as the following assignment problem:

$$\begin{aligned} & \text{minimize } \sum_{i=1}^n \sum_{j=1}^n a_{ij} x_{ij}, \\ & \sum_{i=1}^n x_{ij} = 1, j = 1, \dots, n \\ & \sum_{j=1}^n x_{ij} = 1, i = 1, \dots, n \\ & x_{ij} = \begin{cases} 1, & \text{if job } i \text{ is scheduled in position } j, \\ 0. & \end{cases} \end{aligned} \quad (8)$$

It is well-known that an assignment problem can be solved in  $O(n^3)$  time. For a given vector of  $(k_1, k_2)$ ,  $1 \leq k_1 \leq n_1$ , and  $n_2 \leq k_2 \leq n - 1$ , problem  $P2|fc|TC$  can be solved in  $O(n^3)$  time. There are  $n_1$  possible vectors of  $(k_1, k_2)$ , that is,  $(1, n - 1), \dots, (n_1, n - n_1)$ . Hence, problem  $P2|fc|TC$  can be solved in  $O(n_1 n^3)$  time, since  $n_1 \leq n$ . This implies that the following theorem holds.

**Theorem 3.** Problem  $P2|fc|TC$  can be solved in  $O(n^4)$  time.

*Example 1.* Assume  $N = S_1 \cup S_2$ ,  $S_1 = \{J_1, J_2, J_3\}$ ,  $S_2 = \{J_4, J_5, J_6, J_7, J_8, J_9, J_{10}\}$ ,  $n = 10$ ,  $n_1 = 3$ , and  $n_2 = 7$ , and their processing time is listed in Table 2.

TABLE 2: The 10 jobs processing time.

$i$	1	2	3	4	5	6	7	8	9	10
$p_i$	50	40	30	7	6	5	4	3	2	1

The following solution reports were obtained from Lingo 14.0 package. If  $(k_1, k_2) = (1, 9)$ , then  $TC = 290$ . If  $(k_1, k_2) = (2, 8)$ , then  $TC = 262$ . Therefore, the optimal solution is 262.

The optimal sequences those scheduling on  $M_1$  and  $M_2$  are  $J_3 \rightarrow J_2$  and  $J_{10} \rightarrow J_9 \rightarrow J_8 \rightarrow J_7 \rightarrow J_6 \rightarrow J_5 \rightarrow J_4 \rightarrow J_1$ , respectively.

The Lingo programming code under  $(k_1, k_2) = (2, 8)$  is listed as follows:

**Model:**

Sets:

Jindex/1..10/;

Assign (jindex,jindex):w, x;

Endsets

Data:

```
w = 100 50 400 350 300 250 200 150 100 50
      80 40 320 280 240 200 160 120 80 40
      60 30 240 210 180 150 120 90 60 30
35000 35000 56 49 42 35 28 21 14 7
30000 30000 48 42 36 30 24 18 12 6
25000 25000 40 35 30 25 20 15 10 5
20000 20000 32 28 24 20 16 12 8 4
15000 15000 24 21 18 15 12 9 6 3
10000 10000 16 14 12 10 8 6 4 2
5000 5000 8 7 6 5 4 3 2 1;
```

Enddata

Min = @sum (assign:w\*x);

@for (jindex(i):

@sum (jindex (j): x (I,j)) = 1;

@sum (jindex (j): x (j,i)) = 1;

);

End

**5. Conclusions**

This paper explored two parallel-machine scheduling problems with function constraint. We pointed out that the function constraint occurs not only in the manufacturing system but also in the service system. Therefore, this work is meaningful.

In this research, two important and famous classical objectives, the makespan and the total completion time, were studied. For the makespan problem, we demonstrated that it is NP-hard in the ordinary sense. We presented a polynomial time heuristic for this problem and have proved its worst-case ratio is not greater than  $5/4$ . Computational testing of the heuristic was conducted, and the results revealed that the overall mean percent error of the proposed algorithm is 0.0565%. For the total completion time problem, we have proved that it can be solved in  $O(n^4)$  time.

Future research may examine other topics such as the total load, total tardiness, and number of tardy jobs or extending the model to the uniform parallel-machine setting or other settings.

**Data Availability**

No data were used to support this study.

**Conflicts of Interest**

The authors declare that they have no conflicts of interest.

**References**

- [1] T. Chung, J. N. D. Gupta, H. Zhao, and F. Werner, "Minimizing the makespan on two identical parallel machines with mold constraints," *Computers & Operations Research*, vol. 105, pp. 141–155, 2019.
- [2] D. Xu and D.-L. Yang, "Makespan minimization for two parallel machines scheduling with a periodic availability constraint: mathematical programming model, average-case analysis, and anomalies," *Applied Mathematical Modelling*, vol. 37, no. 14-15, pp. 7561–7567, 2013.
- [3] J. Xu, S.-C. Liu, C. Zhao, J. Wu, W.-C. Lin, and P.-W. Yu, "An iterated local search and tabu search for two-parallel machine scheduling problem to minimize the maximum total completion time," *Journal of Information and Optimization Sciences*, vol. 40, no. 3, pp. 751–766, 2019.
- [4] M. Ji and T. C. E. Cheng, "An FPTAS for parallel-machine scheduling under a grade of service provision to minimize makespan," *Information Processing Letters*, vol. 108, no. 4, pp. 171–174, 2008.
- [5] W.-C. Lee, C.-C. Wu, and P. Chen, "A simulated annealing approach to makespan minimization on identical parallel machines," *The International Journal of Advanced Manufacturing Technology*, vol. 31, no. 3-4, pp. 328–334, 2006.



- [6] Y. Yin, Y. Wang, T. C. E. Cheng, W. Liu, and J. Li, "Parallel-machine scheduling of deteriorating jobs with potential machine disruptions," *Omega*, vol. 69, pp. 17–28, 2017.
- [7] C. Zhao, M. Ji, and H. Tang, "Parallel-machine scheduling with an availability constraint," *Computers & Industrial Engineering*, vol. 61, no. 3, pp. 778–781, 2011.
- [8] W.-H. Kuo and D.-L. Yang, "Parallel-machine scheduling with time dependent processing times," *Theoretical Computer Science*, vol. 393, no. 1–3, pp. 204–210, 2008.
- [9] E. Gerstl and G. Mosheiov, "Scheduling on parallel identical machines with job-rejection and position-dependent processing times," *Information Processing Letters*, vol. 112, no. 19, pp. 743–747, 2012.
- [10] X. Huang and M.-Z. Wang, "Parallel identical machines scheduling with deteriorating jobs and total absolute differences penalties," *Applied Mathematical Modelling*, vol. 35, no. 3, pp. 1349–1353, 2011.
- [11] J. B. Wang and M. Z. Wang, "Minimizing makespan in three-machine flow shops with deteriorating jobs," *Computers & Operations Research*, vol. 40, pp. 47–557, 2013.
- [12] T. C. E. Cheng and C. C. S. Sin, "A state-of-the-art review of parallel-machine scheduling research," *European Journal of Operational Research*, vol. 47, no. 3, pp. 271–292, 1990.
- [13] E. Mokotoff, "Parallel machine scheduling problems: a survey," *Asia-Pacific Journal of Operational Research*, vol. 18, no. 2, pp. 193–242, 2001.
- [14] M. Pfund, J. W. Fowler, and J. N. D. Gupta, "A survey of algorithms for single and multi-objective unrelated parallel-machine deterministic scheduling problems," *Journal of the Chinese Institute of Industrial Engineers*, vol. 21, no. 3, pp. 230–241, 2004.
- [15] S. A. Kravchenko and F. Werner, "Parallel machine problems with equal processing times: a survey," *Journal of Scheduling*, vol. 14, no. 5, pp. 435–444, 2011.
- [16] J. Kaabi and Y. Harrath, "A survey of parallel machine scheduling under availability constraints," *International Journal of Computer and Information Technology*, vol. 3, pp. 238–245, 2014.
- [17] J.-B. Wang and L. Li, "Machine scheduling with deteriorating jobs and modifying maintenance activities," *The Computer Journal*, vol. 61, no. 1, pp. 47–53, 2018.
- [18] M. Pinedo, *Scheduling: Theory, Algorithms, and System*, Prentice-Hall, Upper Saddle River, NJ, USA, 3rd edition, 2008.