Available online at www.sciencedirect.com
ScienceDirect
Omega 37 (2009) 591–599
omega
The International Journal of Management Science
www.elsevier.com/locate/omega

Minimizing number of tardy jobs on a single machine subject to periodic maintenance[☆]

Wen-Jinn Chen[*]

Department of Business Administration, China University of Technology, No. 530, Sec. 3, Zhongshan Rd., Hu-Kou, Hsin-Chu 303, Taiwan, ROC

Received 31 March 2006; accepted 6 January 2008
Available online 11 January 2008

Abstract

This paper considers a single-machine scheduling problem with periodic maintenance. In this study, a schedule consists of several maintenance periods and each maintenance period is scheduled after a periodic time interval. The objective is to find a schedule that minimizes the number of tardy jobs subject to periodic maintenance and nonresumable jobs. Based on the Moore's algorithm, an effective heuristic is developed to provide a near-optimal schedule for the problem. A branch-and-bound algorithm is also proposed to find the optimal schedule. Some important theorems associated with the problem are implemented in the algorithm. Computational results are presented to demonstrate the effectiveness of the proposed heuristic.
© 2008 Elsevier Ltd. All rights reserved.

Keywords: Scheduling; Periodic maintenance; Tardy job; Nonresumable job

1. Introduction

In today's industry, machine breakdown is common for machines running for long periods without maintenance. Such breakdowns make shop behavior hard to predict, and reduce the efficiency of production systems. Maintenance can reduce the breakdown rate with minor sacrifices in production time. The importance of maintenance has been gradually recognized by the decision maker. Therefore, scheduling maintenance in a manufacturing system is a common practice in many companies. When maintenance is performed, the job being processed must be stopped. Consequently, some jobs will be late and several tardy jobs will be generated. Therefore, this study aims to find a schedule with the smallest possible number of tardy jobs that considers jobs and maintenance simultaneously.

In many production systems, periodic inspection, periodic repair and preventive maintenance are usually conducted in the shops. These maintenance works are scheduled regularly or periodically. Therefore, there is a need to develop scheduling method to deal with systems with periodic maintenance.

Scheduling with maintenance is usually treated as availability constraint in the scheduling literature. Some studies have discussed the machine availability problem. For the single-machine problem, Chen [1] solved a scheduling problem with periodic maintenance in which several maintenance intervals are sequenced in a schedule under the nonresumable case. He discussed the problem of minimizing total flow time by assuming several unavailable intervals. Qi et al. [2] considered the same problem in which several unavailable intervals are decision variables.

[☆] This manuscript was processed by Associate Editor Kubiak.
[*] Tel.: +886 3 6991387.
E-mail address: cwj@cute.edu.tw.

0305-0483/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.omega.2008.01.001

Their study attempted to find a schedule with the smallest total completion time. Furthermore, Ji et al. [3] used a different approach for a similar problem. They proved that the worst-case ratio of the classical longest processing time (LPT) algorithm is 2. Chen [4] employed two mixed binary integer programming models to solve a tool-change scheduling problem. In his study, he desired to find a schedule that minimizes the total tardiness subject to tool change time and maximum working time. Adiri et al. [5] assumed that the unavailable time is unknown but with a probabilistic distribution. They distinguished two breakdown cases, i.e., the resumable case and the restart case. By using earliest due date (EDD) and modified shortest processing time (MSPT) rules for the two cases, respectively, the criterion of number of tardy jobs can be minimized. Lee and Liman [6] considered the same problem and provided a proof of NP-hardness. They showed that the tight error bound for the shortest processing time (SPT) heuristic is $\frac{2}{7}$. Lee and Lin [7] stated that the processing time is deterministic, while machine breakdown is a random process following certain distributions. They studied the rate-modifying maintenance problems with objective functions such as expected makespan, total expected completion time, maximum expected lateness, and expected maximum lateness. Pinedo and Rammouz [8] assumed that job processing time is random and job constraints are allowed. Their objectives were to determine the weighted sum of completion time, weighted sum of an exponential function of completion time and weighted number of tardy jobs. For the two-machine problem, Espinouse et al. [9] solved the no-wait flowshop problem for minimizing maximum completion time by assuming each machine may not always be available. Blazewicz et al. [10] studied a similar problem where machines are not available in given time intervals. They provided constructive and local search based algorithms to solve the problem of minimizing makespan. Allahverdi [11] discussed a two-machine problem and showed that if only the first machine breaks down, the LPT policy will minimize the maximum lateness; while if only the second machine breaks down, the SPT policy will be used. Mosheiov [12] considered a two-parallel-machine scheduling problem that minimizes total completion time by assuming each machine is available in a specified interval. Moreover, Schmidt [13] developed an algorithm for finding all jobs of feasible schedules to meet their due dates when each machine is unavailable in a time period. Lee [14–16] provided the researches for the problem that included single and parallel machines with different performance measures such as makespan, total weighted completion time, tardiness, and number of tardy jobs. Kubiak et al. [17] and Sanlaville [18] had investigated this area. For a complete review of availability constraint research, see Schmidt [19] and Lee et al. [20].

In this paper, this study focuses on how to schedule jobs and periodic maintenance, which usually has many maintenance periods. The problem under consideration is to find a schedule that minimizes the number of tardy jobs subject to periodic maintenance and nonresumable jobs.

2. Notation and problem setting

The following notation will be used throughout this paper:

$n=$ number of jobs for processing at time zero.
$J_j=$ job number j.
$M_i=$ maintenance period i.
$p_j=$ processing time of job j.
$d_j=$ due date of job j.
$C_j=$ completion time of job j.
$CM_i=$ completion time of maintenance period i.
$L_j=$ lateness of job j, where $L_j = C_j - d_j$.
$T_j=$ tardiness of job j, where $T_j = \max\{0, L_j\}$.
$n_T=$ number of tardy jobs.
$t=$ amount of time to perform one maintenance.
$T=$ time interval between two maintenance periods.

In addition, $J_{[j]}$ denotes the job placed in the jth position, and $p_{[j]}$, $d_{[j]}$, $C_{[j]}$, $L_{[j]}$, and $T_{[j]}$ are defined accordingly.

This study considers a single-machine scheduling problem in which jobs are nonresumable, i.e., once a job is interrupted it must be restarted from the beginning when a machine becomes available. It is assumed that all jobs are ready for processing at time zero and that, for simplicity, processing times and due dates can take only integer values. For each maintenance, it requires an amount of time t for the performance. There is a time interval T between two consecutive maintenance periods, in which jobs can be scheduled for processing. In this study, a periodic maintenance schedule consists of several maintenance periods and each maintenance period is required after a fixed interval (see Fig. 1). Furthermore, we define a batch as a set of jobs in each interval T. Under these assumptions, a heuristic is developed to minimize the number of tardy jobs with periodic maintenance and nonresumable jobs.

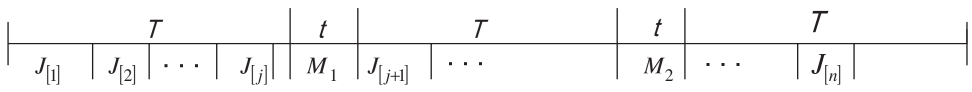

Fig. 1. A schedule with periodic maintenance for nonresumable case.

3. The proposed heuristic

In this section, the proposed heuristic is based on the Moore's algorithm [21], which is used to minimize the number of tardy jobs for the scheduling problem. For completeness, we brief explain the Moore's algorithm in what follows. The algorithm first sequences jobs in EDD rule. Then, the job with the largest processing time preceding the first tardy job in the schedule is rejected from the schedule. The procedure is continued until there are no tardy jobs in the schedule S. The final schedule is obtained by appending the rejected jobs to S. The Moore's algorithm is selected and incorporated into our procedure to create an initial schedule. The following two definitions will be used in the heuristic.

Definition 1. Consider a schedule with several batches. The slack time of a batch is defined as the unscheduled time in a batch.

Definition 2. The potential position is defined as the position immediately after the last position of the first batch where its slack time is greater than or equal to the processing time of the considered job.

The steps of the proposed heuristic are outlined as follows:

Step 1: Obtain a schedule with the smallest number of tardy jobs by the following procedure:

(i) Sequence all jobs by the EDD rule.
(ii) If two jobs have the same due date, place the job with smaller processing time in the front.
(iii) If a tardy job exists, then the job with the largest processing time preceding the first tardy job in the schedule is moved to the tardy job set π. The procedure is continued until there are no tardy jobs in the schedule $S = (J_1, J_2, \ldots, J_n)$.

Step 2: Set $i = 1$, $j = 1$ and $C'_{[0]} = 0$.

Step 3: Generate a periodic maintenance schedule from S by the following procedure:

(i) Compute $C'_{[i]} = C'_{[i-1]} + p_{[i]}$. If $C'_{[i]} \leqslant T$, place $J_{[i]}$ in the batch $B_{[j]}$ and go to Step 3(ii). Otherwise, place $J_{[i]}$ in the next batch $B_{[j+1]}$. Set $C'_{[i]} = p_{[i]}$ and $j = j + 1$.
(ii) Denote by σ_π the periodic maintenance schedule, which is composed of the schedule in all batches and the necessary maintenance periods.
(iii) Calculate $T_{[i]}$ for $J_{[i]}$ in σ_π. If $T_{[i]} > 0$, $J_{[i]}$ is moved to π.
(iv) If $i = n + 1$, stop; otherwise, set $i = i + 1$ and return to Step 3(i).

Step 4: Sequence all jobs in π by the EDD rule.

Step 5: Select a job in π, say J_q. Let σ'_π be the schedule obtained by moving J_q to the potential position of σ_π. If $T_q = 0$ for J_q in σ'_π, go to Step 6. Otherwise, the moving is repeated until all potential positions are checked. Go to Step 7.

Step 6: If the total processing times in all batches are not larger than T, replace σ_π with σ'_π.

Step 7: Repeat Steps 5 and 6 until all jobs in π are checked. Stop.

We now elaborate the steps in detail. In Step 1, Moore's algorithm is applied to create a schedule with the smallest number of tardy jobs. In Step 3(i), a batch is generated, where total processing time for the batch must be smaller than or equal to T. Using Step 3(ii), a periodic maintenance schedule is constructed, where all batches are sequenced and the maintenance periods are incorporated. If a job in σ_π (a periodic maintenance schedule) is tardy, the job is moved from σ_π to π (Step 3(iii)). In Step 5, J_q (a tardy job) is moved to the potential position that may reduce the number of tardy jobs. In Step 6, the batch of jobs in a schedule must be completed in T.

In Step 1, the time complexity of sequencing jobs is $O(n \log n)$. In Step 3, computing $C'_{[i]}$ for $i \in \{1, 2, \ldots, n\}$ requires $O(n)$. Suppose that π with $n_T = n/2$ is generated. The number of remaining jobs in σ_π is $n/2$. Step 4 can be carried out in $O[(n/2)\log(n/2)]$. In Step 5, $T_q = C_q - d_q$ represents the tardiness of J_q. Since $\max_{1 \leqslant q \leqslant (n/2)+1} C_q = C_{(n/2)+1} = \sum_{k=1}^{(n/2)+1} p_{[k]}$, it follows that moving J_q to a potential position and computing T_q can be calculated in a total of $O((n/2)\sum_{k=1}^{(n/2)+1} p_{[k]})$ steps. Additionally, selecting J_q in π for $q \in \{1, 2, \ldots, n/2\}$ requires $O(n/2)$.

Therefore, the overall time complexity of the proposed heuristic is $O(n^2\sum_{i=1}^{n} p_i)$.

4. A numerical example

Example 1. In Table 1 we consider a single-machine scheduling problem with nine jobs, where $T = 8$ and $t = 2$.

In Step 1, we obtain the EDD schedule as given in Table 2. Since J_2 is the first tardy job, J_3 with the largest processing time preceding J_2 is selected (the Moore's algorithm). Then, we move J_3 to π. After the moving, we obtain a schedule with $n_T = 1$ (see Table 3). Applying Step 3, we obtain a periodic maintenance schedule σ_π (see Table 4). There is one maintenance period with $T_2 = 2$. The tardy job J_2 is moved to π. After the moving, we obtain a periodic maintenance schedule with $n_T = 2$. Repeat Step 3 and $T_7 = 1$ is found (see Table 5). Following the same procedure, we produce a periodic maintenance schedule with $n_T = 3$ (see Table 6).

In Steps 5 and 6, we want to reduce the number of tardy jobs. We find that the tardy job J_7 can be moved to the potential position and still satisfy both the conditions in Steps 5 and 6 (see Table 7). This results in the final schedule with $n_T = 2$.

Table 1
The data for Example 1 (in h)

J_i	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9
p_i	1	3	4	5	2	3	2	3	4
d_i	1	14	6	30	10	13	21	10	20

Table 2
The EDD schedule for Example 1 (in h)

J_i	J_1	J_3	J_5	J_8	J_6	J_2	J_9	J_7	J_4
p_i	1	4	2	3	3	3	4	2	5
C_i	1	5	7	10	13	16	20	22	27
d_i	1	6	10	10	13	14	20	21	30

Table 3
The schedule by using the Moore's algorithm (in h)

J_i	J_1	J_5	J_8	J_6	J_2	J_9	J_7	J_4
p_i	1	2	3	3	3	4	2	5
C_i	1	3	6	9	12	16	18	23
d_i	1	10	10	13	14	20	21	30

Table 4
The initial schedule of σ_π for Example 1 (in h)

J_i	J_1	J_5	J_8	M_1	J_6	J_2
p_i	1	2	3		3	3
C_i	1	3	6		13	16
d_i	1	10	10		13	14
T_i	0	0	0		0	2

Table 5
The second schedule of σ_π for Example 1 (in h)

J_i	J_1	J_5	J_8	M_1	J_6	J_9	M_2	J_7
p_i	1	2	3		3	4		2
C_i	1	3	6		13	17		22
d_i	1	10	10		13	20		21
T_i	0	0	0		0	0		1

Table 6
The third schedule of σ_π for Example 1 (in h)

J_i	J_1	J_5	J_8	M_1	J_6	J_9	M_2	J_4
p_i	1	2	3		3	4		5
C_i	1	3	6		13	17		25
d_i	1	10	10		13	20		30
T_i	0	0	0		0	0		0

Table 7
The final schedule of σ_π for Example 1 (in h)

J_i	J_1	J_5	J_8	J_7	M_1	J_6	J_9	M_2	J_4
p_i	1	2	3	2		3	4		5
C_i	1	3	6	8		13	17		25
d_i	1	10	10	21		13	20		30
T_i	0	0	0	0		0	0		0

5. The branch-and-bound algorithm

In this section, a branch-and-bound (B&B) algorithm is presented to provide the optimal schedule for the problem. The algorithm is applied to evaluate the proposed heuristic. The following lemma provides a useful property to solve the problem optimally.

Lemma 1. *Consider two partial schedules composed of the same set of jobs with the same completion time. Then the partial schedule with larger number of tardy jobs is dominated by the other.*

Proof. Denote by S and S' the two considered partial schedules, where S has larger number of tardy jobs. Let R be the partial schedule composed of the remaining jobs. Since S and S' have the same completion time, the number of tardy jobs under the completed schedule

(S, R), i.e., appending R to S, is no smaller than that under (S', R). Thus, S is dominated by S'. This completes the proof. □

Before presenting the proposed B&B algorithm, we state several essential theorems that will be used in the algorithm.

Theorem 1. *Let partial schedule $S = (PS, M_l, \pi_1, J_p, J_q)$, where PS and π_1 are partial schedules. Let S' be obtained from S by interchanging J_p with J_q. If $T_p > 0$ for J_p in both S and S', then S is dominated by S'.*

Proof. Denote by $p\pi_1$ the total processing times of π_1. Then the tardiness of J_p and J_q in S is

$$T_p = CM_l + p\pi_1 + p_p - d_p,$$

$$T_q = CM_l + p\pi_1 + p_p + p_q - d_q.$$

The tardiness of J_q and J_p in S' is

$$T_q = CM_l + p\pi_1 + p_q - d_q,$$

$$T_p = CM_l + p\pi_1 + p_q + p_p - d_p.$$

It is clear that T_q under S' is smaller than T_q under S. This may result in $T_q = 0$ for J_q in S', and $T_q > 0$ for J_q in S. If $T_p > 0$ for J_p in both S and S', then the n_T value under S' is smaller than or equal to that under S. Since S and S' have the same set of jobs with the same completion time, applying Lemma 1, S is dominated by S'. □

Theorem 2. *Let partial schedule $S = (PS, M_l, \pi_1, J_q, J_p)$, where PS and π_1 are partial schedules. Let S' be obtained from S by interchanging J_p with J_q. Denote by $p\pi_1$ the total processing times of π_1. Then S is dominated by S' if both the following conditions are satisfied:*

(i) $CM_l + p\pi_1 + p_q + p_p > d_p$.
(ii) $CM_l + p\pi_1 + p_p \leqslant d_p$.

Proof. Denote by n_TPS and $n_T\pi_1$ the n_T value of PS and π_1, respectively. Then the tardiness of J_q and J_p in S is

$$T_q = CM_l + p\pi_1 + p_q - d_q,$$

$$T_p = CM_l + p\pi_1 + p_q + p_p - d_p.$$

The tardiness of J_p and J_q in S' is

$$T_p = CM_l + p\pi_1 + p_p - d_p,$$

$$T_q = CM_l + p\pi_1 + p_p + p_q - d_q.$$

If condition (i) is satisfied, then the n_T value under S is greater than or equal to $n_TPS + n_T\pi_1 + 1$. If condition (ii) is satisfied, then the n_T value under S' is not greater than $n_TPS + n_T\pi_1 + 1$. Hence, S is dominated by S'. □

Theorem 3. *Let partial schedule $S = (PS, M_l, \pi_1, J_p, \pi_2, J_q)$, where PS, π_1 and π_2 are partial schedules. Let S' be obtained from S by interchanging J_p with J_q. Then S is dominated by S' if both the following conditions are satisfied:*

(i) $p_q - d_q \leqslant p_p - d_p$.
(ii) $d_q \leqslant d_p$.

Proof. Denote by $p\pi_1$ and $p\pi_2$ the total processing times of π_1 and π_2, respectively. Then the tardiness of J_p and J_q in S is

$$T_p = CM_l + p\pi_1 + p_p - d_p,$$

$$T_q = CM_l + p\pi_1 + p_p + p\pi_2 + p_q - d_q.$$

The tardiness of J_q and J_p in S' is

$$T_q = CM_l + p\pi_1 + p_q - d_q,$$

$$T_p = CM_l + p\pi_1 + p_q + p\pi_2 + p_p - d_p.$$

If condition (i) is satisfied, then T_q under S' is not greater than T_p under S. This may result in $T_q = 0$ for J_q in S' and $T_p > 0$ for J_p in S. If condition (ii) is satisfied, then T_p under S' is not greater than T_q under S. Also, this may result in $T_p = 0$ for J_p in S' and $T_q > 0$ for J_q in S. Hence, the n_T value under S' is not greater than that under S. Thus, S is dominated by S'. □

Theorem 4. *Consider two partial schedules composed of the same set of jobs with the same number of tardy jobs. Then the partial schedule with larger completion time is dominated by the other.*

Proof. Denote by S and S' the two assumed partial schedules, where S has larger completion time. Let J_p be any job of the remaining jobs. Since S has larger completion time, the completion time of the partial schedule (S, J_p), i.e., appending J_p to S, is not smaller than that of (S', J_p). It is clear that T_p under (S, J_p) is not smaller than T_p under (S', J_p). This may result in $T_p > 0$ for J_p in (S, J_p) and $T_p = 0$ for J_p in (S', J_p). Since S and S' have the same number of tardy jobs, the number of tardy jobs under (S, J_p) is not smaller than that under (S', J_p). Thus, S is dominated by S'. This completes the proof. □

Table 8
The computational results (data set I)

n	T	t	B&B algorithm Comp. time (s)	Heuristic Comp. time (s)	PED		
					Min.	Mean	Max.
8	10	3	0.07	0.08	0.00	0.00	0.00
		6	0.07	0.08	0.00	0.01	0.03
	15	3	0.06	0.07	0.00	0.00	0.00
		6	0.06	0.06	0.00	0.00	0.00
	20	3	0.05	0.05	0.00	0.00	0.00
		6	0.05	0.05	0.00	0.00	0.00
16	10	3	5.16	0.54	0.00	0.63	5.52
		6	5.24	0.60	0.00	0.76	6.24
	15	3	4.32	0.47	0.00	0.67	4.68
		6	4.73	0.51	0.00	0.58	4.85
	20	3	3.66	0.44	0.00	0.46	3.34
		6	4.16	0.49	0.00	0.55	5.68
24	10	3	84.37	5.47	0.00	0.95	6.70
		6	91.56	5.99	0.00	1.04	8.92
	15	3	82.60	4.82	0.00	0.84	3.96
		6	84.42	5.05	0.00	0.90	6.55
	20	3	79.15	4.15	0.00	0.64	3.79
		6	82.28	4.73	0.00	0.76	5.09
32	10	3	486.40	14.60	0.00	1.06	9.21
		6	499.57	14.24	0.00	1.09	8.65
	15	3	455.06	13.98	0.00	0.93	5.22
		6	463.91	14.09	0.00	0.99	5.36
	20	3	431.68	13.16	0.00	0.74	6.86
		6	445.33	13.88	0.00	0.80	5.21

Table 9
The computational results (data set II)

n	T	t	B&B algorithm Comp. time (s)	Heuristic Comp. time (s)	PED		
					Min.	Mean	Max.
8	10	3	0.08	0.08	0.00	0.00	0.02
		6	0.08	0.09	0.00	0.01	0.05
	15	3	0.06	0.08	0.00	0.00	0.00
		6	0.06	0.07	0.00	0.00	0.00
	20	3	0.05	0.06	0.00	0.00	0.00
		6	0.05	0.06	0.00	0.00	0.00
16	10	3	5.04	0.53	0.00	0.58	5.36
		6	5.36	0.66	0.00	0.77	6.54
	15	3	4.50	0.49	0.00	0.60	3.70
		6	4.76	0.56	0.00	0.63	4.24
	20	3	3.58	0.50	0.00	0.48	4.10
		6	4.00	0.53	0.00	0.50	5.34
24	10	3	82.54	5.51	0.00	0.98	7.28
		6	87.10	5.82	0.00	0.96	8.05
	15	3	79.62	4.94	0.00	0.83	5.68
		6	81.70	5.13	0.00	0.88	5.14
	20	3	77.39	4.36	0.00	0.65	4.73
		6	80.06	4.84	0.00	0.78	4.20
32	10	3	494.14	14.02	0.01	1.06	6.70
		6	502.38	14.47	0.01	1.13	7.36
	15	3	465.07	13.86	0.00	0.94	4.52
		6	482.20	14.15	0.00	0.97	7.20
	20	3	440.15	13.28	0.00	0.78	6.88
		6	458.50	13.90	0.00	0.82	4.30

Table 10
The computational results (data set III)

n	T	t	B&B algorithm Comp. time (s)	Heuristic Comp. time (s)	PED		
					Min.	Mean	Max.
8	10	3	0.08	0.09	0.00	0.01	0.02
		6	0.09	0.10	0.00	0.01	0.04
	15	3	0.07	0.08	0.00	0.00	0.00
		6	0.07	0.08	0.00	0.00	0.00
	20	3	0.05	0.06	0.00	0.00	0.00
		6	0.06	0.06	0.00	0.00	0.00
16	10	3	5.74	1.06	0.00	0.86	7.17
		6	5.82	1.09	0.00	1.02	5.51
	15	3	5.10	0.82	0.00	0.71	5.04
		6	5.31	0.84	0.00	0.76	5.93
	20	3	4.19	0.65	0.00	0.60	4.64
		6	4.38	0.68	0.00	0.64	5.81
24	10	3	95.33	6.09	0.00	1.26	5.46
		6	96.05	6.32	0.00	1.44	9.28
	15	3	88.56	5.40	0.00	1.30	5.30
		6	91.70	5.66	0.00	1.34	6.12
	20	3	82.42	5.09	0.00	1.24	5.15
		6	84.00	5.24	0.00	1.25	6.24
32	10	3	512.79	16.17	0.02	1.57	6.14
		6	526.38	16.88	0.00	1.68	7.52
	15	3	498.17	15.40	0.00	1.31	6.95
		6	506.25	15.93	0.00	1.46	7.16
	20	3	475.63	14.95	0.00	1.24	7.28
		6	482.08	15.22	0.00	1.27	8.30

Table 11
The computational results (data set IV)

n	T	t	B&B algorithm Comp. time (s)	Heuristic Comp. time (s)	PED		
					Min.	Mean	Max.
8	10	3	0.08	0.08	0.00	0.01	0.06
		6	0.09	0.09	0.00	0.01	0.08
	15	3	0.06	0.07	0.00	0.00	0.00
		6	0.08	0.08	0.00	0.00	0.00
	20	3	0.05	0.06	0.00	0.00	0.00
		6	0.05	0.06	0.00	0.00	0.00
16	10	3	5.67	0.98	0.00	0.83	7.10
		6	5.70	1.05	0.00	1.00	7.80
	15	3	5.06	0.78	0.00	0.78	6.58
		6	5.22	0.86	0.00	0.79	7.67
	20	3	4.15	0.57	0.00	0.56	5.63
		6	4.38	0.58	0.00	0.61	5.08
24	10	3	91.52	5.86	0.00	1.19	5.88
		6	93.70	5.94	0.00	1.30	8.76
	15	3	86.11	5.05	0.00	1.21	6.13
		6	88.45	5.36	0.00	1.28	7.82
	20	3	80.34	4.98	0.00	1.07	6.80
		6	83.06	5.14	0.00	1.25	6.38
32	10	3	500.62	15.28	0.00	1.42	8.38
		6	511.09	15.93	0.02	1.56	8.10
	15	3	484.63	14.90	0.00	1.30	7.72
		6	496.40	15.27	0.00	1.34	9.60
	20	3	470.12	14.36	0.00	1.22	8.49
		6	477.74	15.00	0.00	1.28	9.25

For a partial schedule, the lower bound of n_T, denoted by $\underline{n_T}$, is calculated based on a completion of the partial schedule. The complete schedule is constructed by using the Moore's algorithm to sequence the remaining jobs. Then, the steps of the B&B algorithm can be stated as follows:

Step 1: *Initialization.* The schedule of the heuristic is used as the initial schedule of the algorithm.

Step 2: *Branching.* Select a partial schedule with the smallest $\underline{n_T}$ value among all unfathomed partial schedules and let $J_{[i]}$ be the last job of the partial schedule. Produce new partial schedules by placing each of the unscheduled jobs in position $i+1$. It is noted that if the job requires a processing time greater than the slack time, it is scheduled in the next time interval T. If the number of maintenance periods of the new partial schedule is greater than that of the incumbent schedule, then it is eliminated. Use the dominance rules given in Theorems 1–4 and Lemma 1 to eliminate any possible newly created partial schedules.

Step 3: *Bounding.* For each newly created partial schedule, calculate $\underline{n_T}$ by using the proposed scheme stated earlier.

Step 4: *Fathoming.* Fathom those newly created partial schedules whose $\underline{n_T}$ values are no better than that of the incumbent schedule. Update the incumbent schedule if a new partial schedule has a smaller n_T value.

Step 5: *Stop.* If there are nodes remaining unfathomed, go to Step 2; otherwise, stop.

6. Computational results

The effectiveness of the proposed heuristic is evaluated by comparison with optimal schedules obtained by the B&B algorithm. Both the heuristic and the B&B algorithm were coded in Visual BASIC and ran on a PC-686. To demonstrate the heuristic, it was tested by using the data from the experiments. The processing times were randomly generated from a discrete uniform distribution (DU) over [1,10]. The due dates were selected from another DU over $[(1-C-Q/2)\sum_{k=1}^{n} p_{[k]}, (1-C+Q/2)\sum_{k=1}^{n} p_{[k]}]$, with restriction $d_{[i]} \geqslant 0$, where Q and C denote the due date range and tardiness factor, respectively. In our experiments, the same data were used in both the heuristic and the B&B algorithm. The experimental procedure consists of a full factorial design with two settings of Q ($Q = 0.2, 0.6$) and two settings of C ($C = 0.2, 0.6$). For convenience, the four combinations $C=0.2$, $Q=0.2$; $C=0.2$, $Q=0.6$; $C=0.6$, $Q=0.2$; and $C=0.6$, $Q=0.6$ are referred to as data set I, II, III, and IV, respectively. We set three levels of T (10, 15, 20) and two levels of t (3, 6) in the experiments.

The computational results are summarized in Tables 8, 9, 10, and 11 for data sets I, II, III, and IV, respectively. In each complete trial, we randomly drew n ($n = 8, 16, 24, 32$) jobs. The tables provide the information on the average percentage error deviation (PED) and computation time for each combination of n, T and t. The number of testing problems in each n-job trial is 30. We use the PED in tables to validate the performance of the heuristic. The PED is computed as follows: $[(n_T$ by heuristic $- n_T$ by B&B algorithm$)/n_T$ by B&B algorithm$] \times 100$. It can be seen from the tables that problems with smaller C value produce smaller PED and spend less computation time. We also find that problems with larger T value and smaller t value produce smaller PED. This is due to the fact that in this case a smaller number of maintenance periods is generated, and hence a smaller number of potential positions is produced. This results in a n_T value of the heuristic closer to the optimal value. Moreover, the PED increases slightly as the number of jobs increases. The average PED is only 0.72 with a maximum of 1.68. By comparing the heuristic with the B&B algorithm, we observed that the heuristic spends much less computation time than the B&B algorithm. Therefore, the heuristic can be applied for large sized problems, while the B&B algorithm can only be used for small and medium sized problems.

7. Conclusions

This paper examines a single-machine scheduling problem with periodic maintenance. The problem differs from that of other maintenance scheduling problems in that several maintenance periods are considered and each maintenance period is required after an interval. The purpose of this study is to find a schedule that minimizes the number of tardy jobs subject to periodic maintenance and nonresumable jobs. An efficient heuristic has been developed to derive the near-optimal schedule for the problem. The performance of the heuristic is evaluated by comparing its solution with the optimal solution obtained by the B&B algorithm. Several theorems associated with the problem have been implemented in the algorithm. Computational results have indicated that the average computation time is small. In other words, this study can easily derive a feasible schedule in a short time by using the proposed heuristic, even for large problems. In fact, both the heuristic and B&B algorithm perform satisfactorily. Direct application of the results of this study to those companies where maintenance is performed periodically is easy and worthwhile.

References

[1] Chen WJ. Minimizing total flow time in the single-machine scheduling problem with periodic maintenance. Journal of Operational Research Society 2006;57:410–5.

[2] Qi X, Chen T, Tu F. Scheduling the maintenance on a single machine. Journal of Operational Research Society 1999;50:1071–8.

[3] Ji M, He Y, Cheng TCE. Single-machine scheduling with periodic maintenance to minimize makespan. Computers and Operations Research 2007;34:1764–70.

[4] Chen JS. Optimization models for the tool change scheduling problem. Omega 2008;36:888–94.

[5] Adiri I, Frostig E, Rinnooy Kan AHG. Scheduling on a single machine with a single breakdown to minimize stochastically the number of tardy jobs. Naval Research Logistics 1991;38:261–71.

[6] Lee CY, Liman SD. Single machine flow-time scheduling with scheduled maintenance. Acta Informatica 1992;29:375–82.

[7] Lee CY, Lin CS. Single-machine scheduling with maintenance and repair rate-modifying activities. European Journal of Operational Research 2001;135:493–513.

[8] Pinedo MI, Rammouz E. A note on stochastic scheduling machine subject to breakdown and repair. Probability in the Engineering and Informational Sciences 1988;2:41–9.

[9] Espinouse ML, Formanowicz P, Penz B. Minimizing the makespan in the two-machine no-wait flow-shop with limited machine availability. Computers and Industrial Engineering 1999;37:497–500.

[10] Blazewicz B, Breit J, Formanowicz P, Kubiak W, Schmidt G. Heuristic algorithms for the two-machine flowshop with limited machine availability. Omega 2001;29(6):599–608.

[11] Allahverdi A. Two-machine proportionate flowshop scheduling with breakdowns to minimize maximum lateness. Computers and Operations Research 1996;23:909–16.

[12] Mosheiov G. Minimizing the sum of job completion times on capacitated parallel machines. Mathematical and Computer Modeling 1994;20:91–9.

[13] Schmidt G. Scheduling independent tasks with deadlines on semi-identical processors. Journal of the Operational Research Society 1988;39:271–7.

[14] Lee CY. Parallel machines scheduling with non-simultaneous machine available time. Discrete Applied Mathematics 1991;30:53–61.

[15] Lee CY. Machine scheduling with an availability constraint. Journal of Global Optimization 1996;9:395–416.

[16] Lee CY. Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint. Operations Research Letters 1997;20:129–39.

[17] Kubiak W, Blazewicz J, Formanowicz P, Breit J, Schmidt G. Two-machine flow shops with limited machine availability. European Journal of Operational Research 2002;136:528–40.

[18] Sanlaville E. Nearly on line scheduling of preemptive independent task. Discrete Applied Mathematics 1995;57:229–41.

[19] Schmidt G. Scheduling with limited machine availability. European Journal of Operational Research 2000;121:1–15.

[20] Lee CY, Lei L, Pinedo M. Current trends in deterministic scheduling. Annals of Operations Research 1997;70:1–41.

[21] French S. Sequencing and scheduling: an introduction to the mathematics of the job-shop. Chichester, England: Ellis Horwood; 1982.