Applied Mathematical Modelling 34 (2010) 334–342

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

Minimizing the makespan in a single machine scheduling problems with flexible and periodic maintenance

Chinyao Low [a,*], Min Ji [b], Chou-Jung Hsu [a,c], Chwen-Tzeng Su [a]

[a] Institute of Industrial Engineering and Management, National Yunlin University of Science and Technology, Taiwan, ROC
[b] College of Computer Science and Information Engineering, Zhejiang Gongshang University, Hangzhou 310018, PR China
[c] Department of Industrial Engineering and Management, Nan Kai University of Technology, Taiwan, ROC

A R T I C L E I N F O

Article history:
Received 19 January 2009
Received in revised form 17 April 2009
Accepted 30 April 2009
Available online 7 May 2009

Keywords:
Single machine scheduling
Periodic maintenance
Flexible maintenance
NP-hard
Heuristic algorithm

A B S T R A C T

This paper deals with a single machine scheduling problems with availability constraints. The unavailability of machine results from periodic maintenance activities. In our research, a periodic maintenance consists of several maintenance periods. We consider a machine should stop to maintain after a periodic time interval or to change tools after a fixed amount of jobs processed simultaneously. Each maintenance period is scheduled after a periodic time interval. We study the problems under deterministic environment and flexible maintenance considerations. Preemptive operation is not allowed. In addition, we propose a more reasonable flexible model for the real production settings. The objective is to minimize the makespan. The proposed problem is NP-hard in the strong sense and some heuristic algorithms are provided. The purpose is to present an efficient and effective heuristic algorithm so that it will be straightforward and easy to implement. Computational results show that the proposed algorithm first fit decreasing (DFF) performs well.

1. Introduction

Most literature in scheduling assumes that machines are available at all times. However, machines subject to maintenance are found prevalently in process industries and manufacturing systems. In the real world, machines are usually not continuously available [1]. For instance, in the aerospace industry, the machine may not be available in the scheduling period due to a preventive maintenance or tools change for assuring the products keeping high quality. In the PCB manufacture process, drilling machine is one of the most important devices. The machine should stop not only to maintain after a period of processing time but also to change the micro-drill after fixed times of using. The flexible maintenance in the scheduling field means that the maintenance activity has to be started and finished in a predetermined maintenance period (s, t). The time $s(t)$ presents the earliest (latest) time for starting (completion) of the maintenance activity. This scheduling problem is often occurred in the practical manufacturing environments such as computer center, and NC-machine and IC-testing [2]. These stop intervals will affect both the makespan (maximum completion time) of the machine and the total productivity. Therefore, how to schedule to obtain the minimum makespan and enhance the overall performance is an important issue for the industries.

Production scheduling and preventive maintenance planning are the most common and significant problems faced by the manufacturing industry. Some researchers have discussed the machine unavailability problem in the scheduling literature. For the single machine problem, Adiri et al. [3] considered scheduling on a single machine with a single

* Corresponding author.
E-mail address: lowcy@yuntech.edu.tw (C. Low).

0307-904X/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.apm.2009.04.014

breakdown to minimize stochastically the number of tardy jobs. Lee and Liman [4] investigated the single machine scheduling problem of minimizing the sum of job flow-time subject to scheduled maintenance. They provided an NP-completeness proof for the problem. The shortest processing time (SPT) sequence is then shown to have a worst case error bound of 2/7. Sanlaville and Schmidt [5] and Schmidt [6] provided a survey of scheduling problems with limited machine availability. Yang et al. [2] studied problems of a single machine scheduling with a single flexible maintenance activity. The objective was to minimize the makespan. They proved that the problem is NP-hard and provided an efficient heuristic algorithm with complexity $O(n \log n)$ for the proposed problem. Liao and Chen [7] solved a single machine scheduling with periodic maintenance. They proposed a heuristic algorithm for finding the near-optimal solution for large-sized problems. The objective was to minimize the maximum tardiness subject to periodic maintenance and non-resumable constraint. Ji et al. [8] studied a single-machine scheduling with periodic maintenance to minimize makespan. They proved that the worst-case ratio of the classical LPT algorithm is 2, and showed that there is no polynomial time approximation algorithm with a worst-case ratio less than 2 unless $P = NP$, which implies that the LPT algorithm is possibly the best as well. Wu and Lee [9] investigated the scheduling linear deteriorating jobs to minimize makespan with an availability constraint on a single machine. For the sake of simplicity, resumable availability constraint is considered in their paper. They solved that problem by using the 0–1 integer programming technique. Ji et al.[10] researched a topic which is similar to that of Wu and Lee. They considered non-resumable case and proved that both problems are NP-hard under the objective functions minimizing the makespan and the total completion time. Sadfi et al. [11] studied the single machine total completion scheduling problem subject to a period of maintenance. They proposed an approximation algorithm to solve the problem with a worst case error bound of 3/17. Yong et al. [12] considered the problem of scheduling a set of jobs on a single machine on which a rate-modifying activity may be performed. They assumed that the rate-modifying activity can take place only at certain predetermined time points. The objectives were to minimize makespan and total completion time. The analysis shows that the problems are NP-hard even for some special cases. They provided a pseudo-polynomial time optimal algorithm for the problems. Wang et al. [13] studied the problem of scheduling n jobs on a single machine with availability constraints. They considered two intractable special cases, namely, proportional weight case, and single availability constraint case. The objective was to minimize total weighted job completion time. They showed that problem is NP-hard in the strong sense, and proposed two heuristics for these cases and analyzed their worst-case error bounds. Chen [14] studied a single-machine scheduling problem with limited machine availability. The objective was to find a schedule that minimizes the total flow time subject to periodic maintenance and nonresumable jobs. Chen [15,16] studied a single-machine and parallel machine scheduling with flexible and periodic maintenance. He developed several mixed binary integer programming models to solve these problems optimally. And he proposed an efficient heuristic for solving large-sized problems. Chen [17] studied a single machine scheduling problem with periodic maintenance, where the machine is assumed to be stopped periodically for maintenance for a constant time w during the scheduling period. Two mixed binary integer programming (BIP) models are provided for deriving the optimal solution. Additionally, an efficient heuristic is proposed for finding the near-optimal solution for large-sized problems. Xu et al. [18] investigated almost periodic maintenance activities in a parallel machine scheduling problem with minimization of makespan. They showed that the problem is NP-hard, and then an approximation algorithm named BFD-LPT is proposed to solve the problem. Xu et al. [19] showed that there is no polynomial time approximation algorithm with a worst-case performance bound less than 2 unless $P = NP$ for the problem (refers to Chen [17]). The result implies that Chen's heuristic algorithm is the best possible polynomial time approximation algorithm for the considered scheduling problem.

In this article, a more reasonable flexible maintenance model for the real production settings (see Fig. 1) is proposed. Two stratagems that machine should stop to maintain after a periodic time interval or to change tools after a fixed amount of jobs processed are considered simultaneously. We assume that the first time window is arranged in advance. The ith time window depends on the completion time of the $(i-1)$th maintenance task. A maintenance task must be completed within a time window. Each job is processed in the availability time interval t only. Since all of products are produced under well machine conditions, it avoids a risk of product quality problem.

The rest of this article is organized as follows. In the next section, we describe the problem and its complexity. In Section 3, we propose some heuristic algorithms to solve the addressed problem. Section 4 presents the computational experiments and analysis. Section 5 concludes the study with a discussion and future extensions.

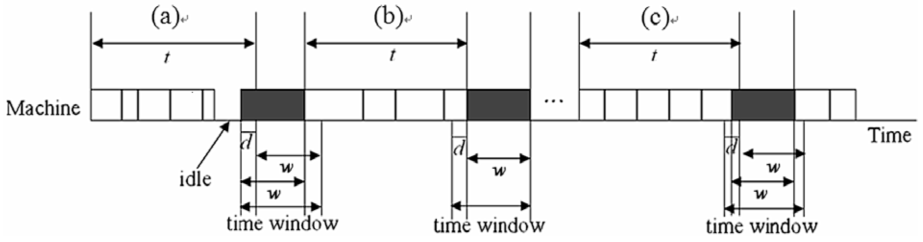

Fig. 1. An illustration for the flexible and periodic maintenance.

2. Problem description and complexity

The following notation was used throughout the study:

n	number of jobs for processing at time zero
J_i	job number i
p_i	the processing time for J_i
t	the length of machine's available interval
k	the maximum number of jobs processed in the machine's available interval
w	the maintenance time of the machine
d	an allowance time, it is a small positive number
S_i	the starting time of the ith maintenance
M_i	the completion time of the ith maintenance
C_i	the completion time of the ith job
W_i	a time window of the ith maintenance task, that is, $W_i = [M_{i-1} + t - d, M_{i-1} + t + w]$

This study considers the single machine scheduling problems with availability constraints. The unavailability of machine results from periodic maintenance activities. In this article, a periodic maintenance consists of several maintenance periods. Two stratagems that a machine should stop to maintain after a periodic time interval or to change tools after a fixed amount of jobs processed are considered simultaneously. A time window is assigned to each maintenance period; that is, the maintenance task must be completed within the designed time window. The flexible maintenance can be discussed in three cases (see Fig. 1). In Fig. 1a, there is an idle time interval. Since the idle time interval is larger than the allowance time d, an idle time interval still exists but it is smaller than the previous idle time interval. In Fig. 1c, although there is an idle time interval, this idle time interval is smaller than the allowance time d. However, under the flexible maintenance consideration, there is not any idle time interval. In Fig. 1b, there is no idle time interval. For a given set of jobs $J = \{J_1, J_2, \ldots, J_n\}$, all jobs are ready for processing at time zero.

For conciseness, the notation of Pinedo [1] is extended here to include resource constraints. This notation includes three fields $\alpha\,|\beta|\gamma$, where α denotes the machine condition, β represents the problem characteristics, and γ is the performance measure requiring optimization. Assume that the "*nprem-pfm*" in the second field denotes the constraints of non-preemptive and the machine with periodic and flexible maintenance. Therefore, the proposed problem is denoted by $1\,|nprmp - pfm|C_{\max}$. It has to be mentioned that the addressed problem is NP-hard in the strong sense (please refer to Chen [17]).

3. Heuristic approaches

The addressed scheduling problem may be regarded as a bin packing process. The bins representing the number of jobs can be assigned for job processing. The objective of the addressed scheduling problem thus becomes a minimizing issue which is to determine the minimum number of bins required to process all jobs.

Our sequence-first, assign-second heuristic approach decomposes the overall problem to exploit each. For the sequencing phase, three rules are considered to form a sequencing priority list: (1) a random method, (2) an arrangement of jobs in decreasing order of corresponding processing time, and (3) an arrangement of jobs in increasing order of corresponding time. Next, for the assignment phase, two fundamental rules, first fit (FF) and best fit (BF) which are widely used for solving a bin packing problem are modified and applied to find an optimal or near-optimal maintenance schedule. That is, there are six combinations of heuristic algorithms. These algorithms are described as follows.

3.1. Heuristics

A. Random order with first fit (RFF)

Step 1: Arrange the jobs in random order of processing times to form a sequencing priority list.
Step 2: Start with the first bin (machine's available interval), and assign the first job to the bin according to the sequencing priority list.
Step 3: Construct a candidate bin set. A job can be taken into a certain bin only if it does not exceed the machine's available interval t or if maximum jobs can be assigned in an interval, k. These conditions can be checked either by the cumulative time for all jobs or number of jobs so far assigned to the certain bin, including the job under consideration. If one of these two conditions is violated, the job cannot be taken into the bin.
Step 4: (a) If the set is empty, a new bin is created, and the job is assigned into the bin; otherwise,
(b) Select the first bin from candidate bin set, and assign the job to the bin.
Step 5: Repeat Steps 3 and 4 until all jobs are assigned.

B. Random order with best fit (RBF)

Step 1: Arrange the jobs in random order of processing times to form a sequencing priority list.

Step 2: Start with the first bin (machine's available interval), and assign the first job to the bin according to the sequencing priority list.

Step 3: Construct a candidate bin set. A job can be taken into a certain bin only if it does not exceed the machine's available interval t or maximum jobs can be assigned in an interval, k. These conditions can be checked either by the cumulative time for all jobs or number of jobs so far assigned to the certain bin, including the job under consideration. If one of these two conditions is violated, the job cannot be taken into the bin.

Step 4: (a) If the set is empty, a new bin is created, and the job is assigned into the bin; otherwise,
(b) Select a bin which has the minimum difference between the machine's available time and the cumulative time for all jobs so far assigned to the certain bin, including the job under consideration from the candidate set, and assign the job to the bin.

Step 5: Repeat Steps 3 and 4 until all jobs are assigned.

C. Decreasing order with first fit (DFF)

Arrange the jobs in decreasing order of processing times to form a sequencing priority list. Next, apply the Steps 2–5 of RFF to complete the job assignments phase.

D. Decreasing order with best fit (DBF)

Arrange the jobs in decreasing order of processing times to form a sequencing priority list. Next, apply the Steps 2–5 of RBF to complete the job assignments phase.

E. Increasing order with first fit (IFF)

Arrange the jobs in increasing order of processing times to form a sequencing priority list. Next, apply the Steps 2–5 of RFF to complete the job assignments phase.

F. Increasing order with best first (IBF)

Arrange the jobs in increasing order of processing times to form a sequencing priority list. Next, apply the Steps 2–5 of RBF to complete the job assignments phase.

It should be noted that the complexity for the first two heuristic algorithms are $O(n)$, and the next heuristics belong to $O(n \log n)$. Those heuristics are coded in Visual BASIC language, and implemented on a Pentium IV personal computer with 2Ghz CPU.

In order to illustrate the proposed heuristic algorithms, an example is provided. Consider a 10-job scheduling problem with flexible periodic maintenance. The job information is listed in Table 1. The length of machine's availability, t, is 20. The maximum number of jobs can be processed before periodic maintenance, k, is 3. The time of a periodic maintenance is 5. The allowance time of a periodic maintenance is 2.

For the addressed example, DFF algorithm is applied to solve the problem. First, arrange the jobs in decreasing order of processing times to form the sequencing priority list which is $\{J_2,J_9,J_{10},J_6,J_5,J_8,J_7,J_4,J_3,J_1\}$. For the job assignment process, J_2 is first picked from the sequencing priority list and assigned to the first available interval of the machine (bin 1). J_1 is the only job that can be selected and assigned to the bin 1 next. The cumulative processing time of bin 1 is thus updated to 19 ($p_2 + p_1 = 19$). Since no job can be assigned to the first available interval, e.g. any job in the list assigned to bin 1 makes the cumulative processing time longer than the predetermined restriction, 20, the second bin is then created. The difference between the cumulative processing time of bin 1 thus far and predetermined processing restriction, 20, is less than 2 (the allowance time of maintenance); the first maintenance task of the machine is started from 19, and completed at 24. This means the starting time of the second available interval for job processing is 24. J_9 is selected from the sequence priority list, and assigned to the certain available interval (bin 2). Since the cumulative processing time of the second available interval is 17, no job in the sequencing priority list can be assigned for processing. In addition, since the difference between the cumulative processing time of bin 2 thus far and predetermined processing restriction is greater than 2, the maintenance is started at time 42 and completed at 47. The arrangement procedure is repeated until all jobs are assigned. The makespan is calculated as 128.

4. The computational experiments

In order to compare with the performances of the six heuristic algorithms, some computational experiments are conducted, and the information used in those experiments is introduced as following:

(1) n is equal to 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, 250, or 300.
(2) p_i is uniformly distributed over [1,20].
(3) w is uniformly distributed over [2,10].

Table 1
Job information for example.

Job_i	1	2	3	4	5	6	7	8	9	10
p_i	2	17	4	7	10	12	9	10	17	14

(4) d is uniformly distributed over [1,3].
(5) k is equal to 8,10,12, or 14.
(6) t is equal to 20, 30, 40, or 50.

To evaluate the performance of the designed heuristics, we have defined that C_{low} is the lower bound on the corresponding makespan and is estimated as follows:

$$C_{\text{low}} = \max\left\{ \sum_{i=1}^{n} p_i + (\lceil n/k \rceil - 1) \times w, \quad \sum_{i=1}^{n} p_i + \left(\left\lceil \sum_{i=1}^{n} p_i / t \right\rceil - 1 \right) \times w \right\}$$

where $\lceil k \rceil$ denotes the smallest integer larger than or equal to k. For each problem, a percentage of error $e = (C_h - C_{low})/C_{low}$ is computed. C_h is the makespan of a heuristic algorithm. We compute the means of all the average percentages of errors for different numbers of jobs. Following the above-mentioned hypothetic numerals, we devise 100 test problems and distribute them to each individual group. Therefore, there are 1300 (13×100) test problems for each problem type and a total of 20,800 ($100 \times 13 \times 4 \times 4$) test problems is generated for the proposed problem $1|nprmp - pfm|C_{\max}$. We also investigate a special case, $k \geqslant n$, which means that consider a machine should stop to maintain after a periodic time interval only. Computational results are shown in Tables 2–5.

According to the results in Tables 2 and 3, we can find that the heuristic DFF is the best heuristic among the six heuristics. The mean percentage error of DFF is 0.012078. This value suggests that the heuristic DFF averagely finds 98.7% of optimality in our schedules.

The special case $k \geqslant n$ means that we consider a machine should stop to maintain after a periodic time interval only. From the results of Tables 4 and 5, we can find the mean percentage error of DFF is 0.007115. It shows that the performance is very good.

We must mention an extremely case, that is, when k is a very small number. We regard it as a tool quality problem. It must be improved from other dimensions, such as materials, production processes, specifications, etc. However, we don't take it into account in this research.

5. Bound analysis

Finally, the asymptotic worst-case performance ratio is designed for evaluating the performance quality of an approximation algorithm. For the addressed problem with minimization of the makespan, let C_{H} denote the makespan of an approximation algorithm H, and C_{OPT} denote the makespan of an optimal off-line algorithm. Then for any instance I, the *worst-case ratio* of algorithm H is defined as the smallest number c such that $C_{\text{H}}(I) \leqslant cC_{\text{OPT}}(I)$, and the asymptotic worst-case ratio of algorithm H is defined as

$$\lim_{C_{\text{OPT}}(I)\to\infty} \sup_{I} \frac{C_{\text{H}}(I)}{C_{\text{OPT}}(I)}.$$

Table 2
Computational results for the heuristics with $p_i \in [1\,20]$, $w \in [6,10]$, and $d \in [1,2]$.

n	k	t	Observed in 1300 test problems					
			RFF	RBF	DFF	DBF	IFF	IBF
N	8	20	0.054398	0.050051	0.025133	0.025613	0.192336	0.192336
		30	0.188561	0.188561	0.025132	0.025613	0.188561	0.188561
		40	0.187351	0.187351	0.025132	0.025613	0.187351	0.187351
		50	0.186900	0.186900	0.025132	0.025613	0.186900	0.186900
	10	20	0.033645	0.030303	0.006090	0.006197	0.317602	0.317602
		30	0.308430	0.308430	0.003522	0.003629	0.308430	0.308430
		40	0.303978	0.303978	0.002648	0.002755	0.303978	0.303978
		50	0.301367	0.301367	0.002340	0.002447	0.301367	0.301367
	12	20	0.015408	0.014651	0.010439	0.010517	0.145912	0.145912
		30	0.128216	0.128216	0.004465	0.004539	0.128216	0.128216
		40	0.117939	0.117939	0.002258	0.002347	0.117939	0.117939
		50	0.112493	0.112493	0.001408	0.001480	0.112493	0.112493
	14	20	0.011172	0.010776	0.033117	0.033127	0.210553	0.210553
		30	0.179784	0.179784	0.015084	0.015092	0.179784	0.179784
		40	0.164308	0.164308	0.007501	0.007512	0.164308	0.164308
		50	0.155519	0.155519	0.003841	0.003870	0.155519	0.155519
Mean			0.153092	0.152539	0.012078	0.012248	0.200078	0.200078

N is equal to 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, 250, or 300.

Table 3
Computational results for the heuristics with $p_i \in [1,20], w \in [6,10]$, and $d \in [1,2]$.

t	k	n	Observed in 1600 test problems					
			RFF	RBF	DFF	DBF	IFF	IBF
T	K	20	0.142478	0.141757	0.014326	0.014809	0.177689	0.177689
		30	0.141251	0.140689	0.010465	0.010772	0.181452	0.181452
		40	0.149085	0.148573	0.011863	0.012056	0.192983	0.192983
		50	0.17157	0.170536	0.014418	0.014591	0.21971	0.21971
		60	0.170017	0.169353	0.01171	0.011984	0.219832	0.219832
		70	0.151277	0.150864	0.010393	0.010537	0.198263	0.198263
		80	0.147022	0.146661	0.01033	0.01045	0.193765	0.193765
		90	0.152025	0.151394	0.010338	0.010501	0.199786	0.199786
		100	0.150679	0.150253	0.011173	0.011279	0.198622	0.198622
		150	0.149861	0.149313	0.012233	0.012309	0.198652	0.198652
		200	0.147662	0.147309	0.012317	0.012382	0.197007	0.197007
		250	0.149213	0.148833	0.013211	0.013267	0.199467	0.199467
		300	0.168055	0.167478	0.014233	0.014286	0.223789	0.223789
Mean			0.153092	0.152539	0.012078	0.012248	0.200078	0.200078

T is equal to 20, 30, 40, or 50.
K is equal to 8, 10, 12, or 14.

Table 4
Computational results for the heuristics with $p_i \in [1,20]$, $w \in [6,10]$, and $d \in [1,2]$.

t	n	Observed in 400 test problems					
		RFF	RBF	DFF	DBF	IFF	IBF
T	20	0.05293	0.050199	0.014681	0.0151	0.17895	0.17895
	30	0.041505	0.038717	0.010332	0.010557	0.179785	0.179785
	40	0.045969	0.042196	0.010658	0.010948	0.201908	0.201908
	50	0.033629	0.03144	0.007422	0.007578	0.179509	0.179509
	60	0.032272	0.030142	0.009127	0.009224	0.18034	0.18034
	70	0.029032	0.025936	0.006808	0.006894	0.178477	0.178477
	80	0.025039	0.023629	0.005391	0.005528	0.180201	0.180201
	90	0.023871	0.022633	0.005965	0.006088	0.180545	0.180545
	100	0.03074	0.02716	0.00655	0.006631	0.205329	0.205329
	150	0.024104	0.021594	0.004657	0.004767	0.201663	0.201663
	200	0.015316	0.013962	0.003614	0.003697	0.175853	0.175853
	250	0.013624	0.012297	0.00339	0.003435	0.178706	0.178706
	300	0.017214	0.014892	0.003901	0.003962	0.203352	0.203352
Mean		0.029634	0.027292	0.007115	0.007262	0.186509	0.186509

T is equal to 20, 30, 40, or 50.

Table 5
Computational results for the heuristics with $p_i \in [1,20]$, $w \in [6,10]$, and $d \in [1,2]$.

n	t	Observed in 1300 test problems					
		RFF	RBF	DFF	DBF	IFF	IBF
N	20	0.056525	0.051752	0.025126	0.025568	0.190549	0.190549
	30	0.034431	0.031225	0.002491	0.002566	0.304429	0.304429
	40	0.016987	0.015932	0.000637	0.000716	0.107363	0.107363
	50	0.010594	0.01026	0.000206	0.000199	0.143696	0.143696
Mean		0.029634	0.027292	0.007115	0.007262	0.186509	0.186509

N is equal to 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, 250, or 300.

If we assume $d = 0$, then the proposed problem can be formulated as a k-item bin-packing problem (kBP). It is directly derived from bin-packing problem by adding the constraint that at most k items can be packed into every bin. The first heuristics for (kBP) were analyzed by Krause et al. [20]. They considered a system there are k processors that share a common memory of a fixed capacity. A set of tasks are to be scheduled on these processors, and each task has a corresponding processing and certain memory requirement. Obviously, the aim of the system is to execute all tasks in the shortest possible time. In the off-line issue of the problem, they introduced several approximation algorithms and shown that all algorithms have an asymptotic worst-case performance ratio of 2. For the DFF algorithm, they proved that $B_{\mathrm{DFF}}(I)$ is bounded by $B_{\mathrm{DFF}}(I) < (2 - 2/k)B_{\mathrm{OPT}}(I) + 1$ under the condition of $k \geqslant 2$, where $B_{\mathrm{DFF}}(I)$ denote the number of bins used by heuristic DFF and $B_{\mathrm{OPT}}(I)$ denote the minimum (optimum) number of bins required to pack list I.

Similar to Ji et al. [8], if a time interval between two consecutive maintenance activities is treated as a batch (bin) for the addressed problem, then the longest processing time (LPT) algorithm is equivalent to the DFF algorithm. A schedule π can be denoted as $\pi = (B_1, w, B_2, w, \ldots, B_{L-1}, w, B_L)$, where L is the number of batches and B_i is the ith batch of the jobs. Let $l(B_i)$ denote the total processing time of the ith batch of the jobs. If $l(B_i)$ is less than $t - d$, then set $l(B_i)$ as $t - d$ is under flexible maintenance consideration. That is, $l(B_i) \in [t - d, t]$. We first present some properties which are all straightforward.

Property 1 (Ji et al. [8, p. 1766]). *The optimal schedule must have the minimum number of batches, i.e. it corresponds to an optimal solution for the bin-packing problem.*

Property 2. *The flexible maintenance consideration does not affect the number of batches which are produced by a heuristic H on list L.*

Proof. The result is derived from the definition of the model. □

Lemma 1. *$B_{DFF}(I) \leqslant 2B_{OPT}(I)$.*

Proof. From Theorem 2.2 of Krause et al. [18], we have $B_{\mathrm{DFF}}(I) < (2 - 2/k)B_{\mathrm{OPT}}(I) + 1 = (B_{\mathrm{OPT}}(I) + 1) + (1 - 2/k)B_{\mathrm{OPT}}(I)$. Due to $B_{\mathrm{DFF}}(I)$ and $B_{\mathrm{OPT}}(I)$ are all positive integers, so we obtain $B_{\mathrm{DFF}}(I) < (B_{\mathrm{OPT}}(I) + 1) + (1 - 2/k)B_{\mathrm{OPT}}(I) \leqslant (B_{\mathrm{OPT}}(I) + 1) + (B_{\mathrm{OPT}}(I) - 1) = 2B_{\mathrm{OPT}}(I)$. □

Lemma 2

$$B_{\mathrm{DFF}}(I)/(B_{\mathrm{OPT}}(I) - 1) \leqslant \begin{cases} 2, & \text{if } k = 2 \\ 3, & \text{if } k = 3 \\ 4, & \text{if } k \geqslant 4 \end{cases}.$$

Proof. It is trivial that $B_{\mathrm{OPT}}(I) \geqslant 2$. If $k = 2$, we have $B_{\mathrm{DFF}}(I) < (2 - 2/k)B_{\mathrm{OPT}}(I) + 1 = B_{\mathrm{OPT}}(I) + 1$. So we have $B_{\mathrm{DFF}}(I) = B_{\mathrm{OPT}}(I)$ due to $B_{\mathrm{DFF}}(I)$ and $B_{\mathrm{OPT}}(I)$ are all positive integers. Therefore, we obtain $B_{\mathrm{DFF}}(I)/(B_{\mathrm{OPT}}(I) - 1) = 1 + 1/(B_{\mathrm{OPT}}(I) - 1) \leqslant 2$ since $B_{\mathrm{OPT}}(I) \geqslant 2$.

If $k = 3$, then we have $B_{\mathrm{DFF}}(I) < (2 - 2/k)B_{\mathrm{OPT}}(I) + 1 = 4B_{\mathrm{OPT}}(I)/3 + 1$. We first assume $B_{\mathrm{OPT}}(I) \geqslant 3$. thus we obtain $B_{\mathrm{DFF}}(I)/(B_{\mathrm{OPT}}(I) - 1) < 4/3 + 7/[3(B_{\mathrm{OPT}}(I) - 1)] \leqslant 2.5$ due to $B_{\mathrm{OPT}}(I) \geqslant 3$. If $B_{\mathrm{OPT}}(I) = 2$, we have $B_{\mathrm{DFF}}(I) < 4B_{\mathrm{OPT}}(I)/3 + 1 = 11/3$. Therefore, we get $B_{\mathrm{DFF}}(I) \leqslant 3$ due to $B_{\mathrm{DFF}}(I)$ is positive integer. So we conclude that $B_{\mathrm{DFF}}(I)/(B_{\mathrm{OPT}}(I) - 1) \leqslant 3$.

If $k \geqslant 4$, then from Lemma 1, we have $B_{\mathrm{DFF}}(I)/(B_{\mathrm{OPT}}(I) - 1) \leqslant 2 + 2/(B_{\mathrm{OPT}}(I) - 1) \leqslant 4$ due to $B_{\mathrm{OPT}}(I) \geqslant 2$. □

Theorem 1. *The worst-case ratio of DFF algorithm for problem $1|nprmp - pfm|C_{\max}$ is 1 for*

$$k = 1, \text{and not greater than} \begin{cases} 2(t + w)/(t + w - d), & \text{if } k = 2 \\ 3(t + w)/(t + w - d), & \text{if } k = 3 \\ 4(t + w)/(t + w - d), & \text{if } k \geqslant 4 \end{cases}.$$

Proof. We first claim that $k \geqslant 2$. Otherwise, we have $C_{\mathrm{DFF}}(I) = C_{\mathrm{OPT}}(I)$. From the definition of the problem and the rule of DFF algorithm, we can easily obtain $C_{\mathrm{DFF}}(I) \leqslant B_{\mathrm{DFF}}(I)(t + w)$. On the other hand, from Properties 1 and 2, we have $C_{\mathrm{OPT}}(I) \geqslant (B_{\mathrm{OPT}}(I) - 1)(t + w - d)$. Therefore, from Lemma 2, we can easily get the conclusions. □

Let the total processing times of the jobs in the last batch for the optimal schedule and the DFF schedule be x and y, respectively. Let the amount of the reduced idle time of each batch i for the optimal schedule and of each batch j for the DFF schedule under flexible maintenance consideration be e_i and f_j, i.e.,$e_i = t - l(B_i) \in [0, d]$ and $f_j = t - l(B_j) \in [0, d]$.

Theorem 2. *The asymptotic worst-case performance ratio of DFF algorithm for problem $1|nprmp - pfm|C_{max}$ is 1 for k = 1 and $2 - \frac{2}{k}$ for $k \geqslant 2$.*

Proof. We first claim that $k \geqslant 2$. Otherwise, we have $C_{\mathrm{DFF}}(I) = C_{\mathrm{OPT}}(I)$.

The definition of the model implies that

$$C_{\mathrm{DFF}}(I) = (B_{\mathrm{DFF}}(I) - 1)(t + w) + x - \sum_{j=1}^{B_{\mathrm{DFF}}(I)-1} f_j$$

and

$$C_{\mathrm{OPT}}(I) = (B_{\mathrm{OPT}}(I) - 1)(t + w) + y - \sum_{i=1}^{B_{\mathrm{OPT}}(I)-1} e_i.$$

From Property 1, the optimal schedule must have the minimum amount of the total idle times of the batches. This implies that the amount of the total idle times of the optimal schedule must be bounded by a finite number. Therefore, the total amount of the reduced idle times of the optimal schedule under flexible maintenance consideration $(\sum_{i=1}^{B_{\text{OPT}}(I)-1} e_i)$ must be bounded by a finite number as well, say z.

From Property 2 and the results proposed by Krause et al. [18], the inequality $B_{\text{DFF}}(I) < (2-2/k)B_{\text{OPT}}(I) + 1$ also holds. Since

$$C_{\text{DFF}}(I) = (B_{\text{DFF}}(I)-1)(t+w) + x - \sum_{j=1}^{B_{\text{DFF}}(I)-1} f_j \leqslant (B_{\text{DFF}}(I)-1)(t+w) + x$$

and

$$B_{\text{DFF}}(I) < (2-2/k)B_{\text{OPT}}(I) + 1 \quad \text{then} \quad C_{\text{DFF}}(I) < (2-2/k)B_{\text{OPT}}(I)(t+w) + x.$$

We have

$$\frac{C_{\text{DFF}}(I)}{C_{\text{OPT}}(I)} < \frac{(2-2/k)B_{\text{OPT}}(t+w)+x}{C_{\text{OPT}}(I)} \leqslant \frac{(2-2/k)C_{\text{OPT}}(I) + (2-2/k)(t+w) + x - y + z}{C_{\text{OPT}}(I)}.$$

Thus,

$$\lim_{C_{\text{OPT}}(I)\to\infty} \frac{C_{\text{DFF}}(I)}{C_{\text{OPT}}(I)} \leqslant (2-2/k) + \lim_{C_{\text{OPT}}(I)\to\infty} \frac{(2-2/k)(t+w)+x-y+z}{C_{\text{OPT}}(I)} = 2 - \frac{2}{k}. \qquad \square$$

6. Conclusions

To improve the overall performances, the single machine scheduling problem becomes an important issue. In this article, we investigate a single machine problem with two maintenance stratagems simultaneously and show that this problem is NP-hard in the strong sense. Under the concepts of the NP-hard in the strong sense, we propose six kinds of heuristic algorithms. The computational experiments show that the performance of the proposed algorithms DFF is quite satisfactory. A special case,$k \geqslant n$, means we consider that the machine should stop to maintenance after a period available interval t only. The performance of the proposed algorithms DFF is also satisfactory.

In order to further obtain the satisfying result and develop the practical heuristic algorithms, it is suggested that we should simultaneously take other performance measurement factors, such as the total completion time or maximum lateness into account. Moreover, another problem derived from this study, such as the m parallel machines issue, may need to be addressed.

Acknowledgements

We thank two anonymous reviewers for their constructive and helpful comments on an earlier version of our paper. Ji was supported in part by the National Natural Science Foundation of China under Grant number 10801121, and the Science Fund for Young Scholars of Zhejiang Gongshang University under Grant number Q09-21.

References

[1] M. Pinedo, Scheduling Theory, Algorithms, and Systems, Prentice-Hall, New Jersey, 2002.
[2] D.L. Yang, C.L. Hung, C.J. Hsu, M.S. Chern, Minimizing the makespan in a single machine scheduling problem with a flexible maintenance, Journal of the Chinese Institute of Industrial Engineers 19 (2002) 63–66.
[3] I. Adiri, E. Frostig, A.H.G. Rinnooy Kan, Single machine flow-time scheduling with a single breakdown to minimize stochastically the number of tardy jobs, Naval Research Logistics 38 (1991) 261–271.
[4] C.Y. Lee, S.D. Liman, Single machine flow-time scheduling with scheduled maintenance, Acta Informatica 29 (1992) 375–382.
[5] E. Sanlaville, G. Schmidt, Machine scheduling with availability constraints, Acta Informatica 9 (1998) 795–811.
[6] G. Schmidt, Scheduling with limited machine availability, European Journal of Operational Research 121 (2000) 1–15.
[7] C.J. Liao, W.J. Chen, Single-machine scheduling with periodic maintenance and nonresemable jobs, Computers and Operations Research 30 (2003) 1335–1347.
[8] M. Ji, Y. He, T.C.E. Cheng, Single-machine scheduling with periodic maintenance to minimize makespan, Computers and Operations Research 34 (2007) 1764–1770.
[9] C.C. Wu, W.C. Lee, Scheduling linear deteriorating jobs to minimize makespan with an availability constraint on a single machine, Information Processing Letters 87 (2003) 89–93.
[10] M. Ji, Y. He, T.C.E. Cheng, Scheduling linear deteriorating jobs with an availability constraint on a single machine, Theoretical Computer Science 362 (2006) 115–126.
[11] C. Sadfi, B. Penz, C. Rapine, J. Blazewicz, P. Formanowicz, An improved approximation algorithm for the single machine total completion time scheduling problem with availability constraints, European Journal of Operational Research 161 (2005) 3–10.
[12] H. Yong, M. Ji, T.C.E. Cheng, Single machine scheduling with a restricted rate-modifying activity, Naval Research Logistics 52 (2005) 361–369.
[13] G. Wang, H. Sun, C. Chu, Preemptive scheduling with availability constraints to minimize total weighted completion times, Annals of Operations Research 133 (2005) 183–192.
[14] W.J. Chen, Minimizing total flow time in the single-machine scheduling problem with periodic maintenance, Journal of the Operational Research Society 57 (2006) 410–415.
[15] J.S. Chen, Single-machine scheduling with flexible and period maintenance, Journal of the Operational Research Society 57 (2006) 703–710.

[16] J.S. Chen, Optimization models for the machine scheduling problem with a single flexible maintenance activity, Engineering Optimization 38 (2006) 53–71.
[17] J.S. Chen, Scheduling of nonresumable jobs and flexible maintenance activities on a single machine to minimize makespan, European Journal of Operations Research 190 (2008) 90–102.
[18] D. Xu, K. Sun, H. Li, Parallel machine scheduling with almost periodic maintenance and non-preemptive jobs to minimize makespan, Computers and Operations Research 35 (2008) 1344–1349.
[19] D. Xu, K. Sun, H. Li, A note on scheduling of nonresumable jobs and flexible maintenance activities on a single machine to minimize makespan, European Journal of Operations Research 197 (2009) 825–827.
[20] K.L. Krause, V.Y. Shen, H.D. Schwetman, Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems, Journal of the Association for Computing Machinery 22 (1975) 522–550.