Computers & Operations Research 38 (2011) 755–770

Contents lists available at ScienceDirect

Computers & Operations Research

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

Minimizing total completion time on a single machine with a flexible maintenance activity

Shan-lin Yang [a,c], Ying Ma [a,b,*], Dong-ling Xu [a,b], Jian-bo Yang [a,b]

[a] School of Management, Hefei University of Technology, 193 Tunxi Road, Hefei 230009, Anhui, China
[b] Manchester Business School, The University of Manchester, Manchester M15 6PB, UK
[c] Key Laboratory of Process Optimization and Intelligent Decision-making, Ministry of Education, Hefei 230009, Anhui, China

ARTICLE INFO

Available online 16 October 2010

Keywords:
Single machine scheduling
Flexible maintenance
Shortest processing time
Dynamic programming
Branch-and-bound

ABSTRACT

A problem of jointly scheduling multiple jobs and a single maintenance activity on a single machine with the objective of minimizing total completion time is considered in this paper. It is assumed that the machine should be stopped for maintenance which takes a constant duration within a predefined period. The problem is generalized from the one with a fixed maintenance in that it relaxes the starting time of the maintenance from a fixed time point to a predefined period. Both resumable and nonresumable cases are studied. First, three properties of an optimal solution to each of the two cases are identified. Then it is shown that the proposed shortest processing time (SPT) algorithm is optimal for the resumable case. As for the nonresumable case, the conditions under which the SPT algorithm is optimal are also specified. Furthermore, it is shown that relaxing the starting time of the maintenance cannot improve the relative error bound of the SPT algorithm. The focus of the paper is presented afterwards, which is to develop a dynamic programming algorithm and a branch-and-bound algorithm to generate an optimal solution for this case. Experimental results show that these algorithms are effective and complementary in dealing with different instances of the problem.

Statement of scope and purpose: In the majority of scheduling problems with preventive maintenance, maintenance periods are assumed to be constant. However, in real industry settings, such periods may be flexible. Therefore, it is necessary to consider scheduling problems with flexible maintenance. This paper focuses on a single machine problem in which job processing and machine maintenance have to be scheduled simultaneously. The objective is to minimize total completion time of jobs for both resumable and nonresumable cases. For the resumable case, a SPT algorithm proposed in this paper is shown to be optimal. On the other hand, for the nonresumable case, the relative worst-case error bound of the SPT algorithm is analyzed, and further, a dynamic programming algorithm and a branch-and-bound algorithm are proposed to solve this problem optimally. Finally, experimental results are provided to show the effectiveness and complementarity of the above algorithms.

1. Introduction

Scheduling problems with preventive maintenance have received increasing attention in the last decade, as the importance of the applications has been recognized. There are two types of preventive maintenance: deterministic maintenance and flexible maintenance. The former means that maintenance periods are determined before jobs are scheduled, i.e., the starting times of the maintenance activities are fixed in advance. Scheduling with this type of maintenance is often referred in literature as "scheduling with availability constraints", "scheduling with limited machine availability" or "scheduling with fixed non-availability intervals" because during maintenance periods the machine is unavailable for processing any job. The latter means that the schedule of maintenance periods are determined jointly with the schedule of jobs, i.e., the starting times of the maintenance activities are allowed to be flexible and are determined in the scheduling process. This problem is known as "scheduling with flexible maintenance", "scheduling with unfixed availability constraints", "integrating preventive maintenance planning and production scheduling", or "(simultaneously/jointly) scheduling jobs and maintenance activities".

The first type of maintenance, the deterministic one, has been addressed by significant number of works. Comprehensive reviews were provided by Lee et al. [1], Sanlaville and Schmidt [2], Schmidt [3], and Ma et al. [4]. The following is a review of the works dealing with a subset of the problems, i.e., the single machine total (weighted) completion time problems with only one maintenance activity.

* Corresponding author at: School of Management, Hefei University of Technology, 193 Tunxi Road, Hefei 230009, Anhui, China. Tel.: +86 551 2901501; fax: +86 551 2905263.
E-mail address: maying_0102@yahoo.cn (Y. Ma).

0305-0548/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2010.09.003

As far as the total completion time of jobs is concerned, while the resumable case can be solved optimally by the shortest processing time (SPT) rule, the corresponding nonresumable case is NP-hard [5] and the SPT rule leads to a relative worst-case error bound of 2/7 [6], where the resumable and nonresumable case are defined by Lee [7]: a job is resumable if it cannot be finished before the unavailable period of a machine and can continue after the machine is available again; on the other hand, it is nonresumable if it has to restart, rather than continue after the machine becomes available.[1] In addition, an approximation algorithm with a tight error bound of 3/17, a parametric $O(n\log n)-$ algorithm with better error bounds and a polynomial time approximation scheme (PTAS) were proposed by Sadfi et al. [8], Breit [9], and He et al. [10], respectively.

When the objective is to minimize the total weighted completion time of jobs, Lee [7] showed that both the resumable and nonresumable cases are NP-hard and the relative error bound of weighted shortest processing time (WSPT) rule can be arbitrarily large even if the weights are proportional to the processing times. For the resumable case, he also developed a combine-algorithm which combines the WSPT and the largest weight rule and showed that its error bound can be arbitrarily large, but when the weights are proportional to the processing times, it leads to a tight error bound of 1. In the paper, a dynamic programming model was also proposed to solve the problem optimally. Furthermore, a heuristic with a tight relative error bound of 1 was proposed by Wang et al. [11] to solve the same problem. In the case that the jobs are nonresumable, Kacem and Chu [12] showed that both WSPT and MWSPT (modified WSPT) have a tight relative error bound of 2 under some conditions. Then, Kacem [13] proposed a 2-approximation algorithm and showed that this bound is tight. Three exact algorithms, a branch-and-bound, a mixed-integer linear programming and a dynamic programming, were proposed by Kacem et al. [14] to solve this problem optimally. Later, new properties and lower bounds were proposed to improve the branch-and-bound algorithm by Kacem and Chu [15].

Besides the literature on the first type of maintenance mentioned above, there are also a few works dealing with the second type of maintenance. Since the maintenance is flexible, most of the works are concerned with nonresumable cases. So, in the subsequent discussions, jobs are assumed to be nonresumable, unless otherwise specified. Based on practical requirements, various kinds of flexible maintenance are considered.

The first kind is that a machine must be maintained after working continuously for a period of time and the maximum allowed continuous working time is fixed. This model were considered by Mosheiov and Sarig [16,17], Lee and Chen [18], Levin et al. [19], Allaoui et al. [20], Qi et al. [21], Qi [22], Sbihi and Varnier [23], and Graves and Lee [24]. In some of those papers, it is assumed that each machine is maintained only once [16–20], while in others, machines must be maintained several times. Meanwhile, the same scheduling model was independently studied by Akturk et al. [25,26] with a different motivation from tool management point of view where maintenance activities are interpreted as tool changes.

The second kind is that each maintenance activity corresponds to a predefined time interval, in which the maintenance activity can be scheduled. Yang et al. [27] were the first to study problem with this kind of maintenance. They considered a single machine makespan problem with only one maintenance activity, demonstrated that the proposed problem is NP-hard and provided some solvable cases and a heuristic. When there are multiple maintenance activities on a single machine, Chen [28] developed four mixed binary integer programming (BIP) models and a heuristic to solve the mean flow time problem. Four problems based on different job characteristics and objective functions are addressed by Lau and Zhang [29]. In the paper, the complexity of these problems and approximation ratio of some proposed heuristics are analyzed. In another paper by Chen [30], both resumable and nonresumable cases were considered for single machine and parallel machine scheduling with the objective of minimizing the total tardiness. To solve them, eight mixed BIP models and a heuristic were proposed. Later, Chen [31] extended his research to the single machine makespan problem, proposed two mixed BIP models and a heuristic, whose worst-case performance bound was showed to be 2 by Xu et al. [32] later. Xu et al. [33] studied a parallel machine scheduling problem where each machine setting is the same as that in Ref. [31]. The objective is to minimize the completion time of the last finished maintenance. They showed that the problem is NP-hard, and unless P=NP, there is no polynomial time ρ—approximation algorithm for this problem for any $\rho<2$. Then they proposed a polynomial time approximation algorithm to solve the problem. When a flexible job shop is considered, Gao et al. [34] proposed a hybrid genetic algorithm to minimize the following three objectives: makespan, maximal machine workload and total workload of the machines.

Other kinds of flexible maintenance were also considered by researchers. For example, Lau and Zhang [29] assumed that a fixed number of jobs must be completed within an available interval; Dell' Amico and Martello [35], Liao et al. [36], and Yang et al. [37] assumed that the machine must be maintained after completing a fixed number of jobs at most; Low et al. [38] assumed two stratagems that a machine should stop to maintain after a periodic time interval or to change tools after completing a fixed number of jobs; Xu et al. [39] assumed that the length of the time interval between any two consecutive maintenance activities is between two given positive numbers in a parallel machine setting, i.e., it belongs to the second kind of maintenance except that the time needed to perform one maintenance activity is increasing linearly instead of being fixed; Cassady and Kutanoglu [40,41], Sortrakul et al. [42], and Ruiz et al. [43] assumed that the time to failure of a machine is governed by a Weibull probability distribution; Berrichi et al. [44], Kaabi et al. [45], Youssef et al. [46], and Chen and Liao [47] defined other different models.

The flexible maintenance model discussed in this paper belongs to the second kind, i.e., during a scheduling period a single machine should be stopped within a predefined time period for maintenance which takes a fixed time to complete. Detailed problem description is given in the next section. To the best knowledge of the authors, the problem has not been dealt with previously with the objective of minimizing the total completion time. This maintenance model is originated from an air-conditioner manufacturer where the second author worked for three years. In this company, there are many types of machines, such as aluminium foil puncher, copper tube bender, copper tube expander, heat exchanger bender, degreasing furnace, drying furnace, synthetic test equipment, fluoride injector, etc. Some simple maintenance of each machine can be done by shop floor workers, such as cleaning, lubricating, fixing screw, and spot inspection. But the more difficult maintenance has to be done by a serviceman from an equipment company affiliated to the same group, such maintenance include: the maintenance of hydraulic system, circuit system and punching program, the replacement of the punching die, the adjustment of expending rod, flanging height and bending angle, and the

[1] Actually, a nonresumable case is the same as a nonpreemptive one, i.e., a job must be completed without interruption once started. Therefore, without loss of generality, nonresumable case and nonpreemptive case are used interchangeably in this paper.

systemic maintenance, etc. In order to minimize disruption to production and to obtain more economic benefit for the whole group, the serviceman and the air-conditioner manufacturer will collaborate to find a maintenance period. At first, the serviceman provides the manufacturer a time period (longer than the required time for the maintenance) in advance according to the predefined general maintenance plan (annual plan, monthly plan or daily plan). Then the manufacturer simultaneously schedule jobs and maintenance under this constraint. Once the maintenance period is scheduled, the serviceman will be informed of the time to conduct the maintenance.

The rest of this paper is organized as follows. The problem is described in Section 2. In Section 3, basic properties of an optimal solution are presented, based on which, the SPT algorithm and its performance in dealing with resumable and nonresumable cases are studied in Section 4. Section 5 and Section 6 are aimed at proposing a dynamic programming algorithm and a branch-and-bound algorithm. Computational experiments are then given in Section 7 to demonstrate the effectiveness and complementarity of these algorithms, followed by the conclusions and discussions on future research in Section 8.

2. Problem description

Suppose a single machine should be stopped within a predefined time period $[s, t]$ for maintenance that takes a fixed time r ($r \le t-s$) to complete. The time $s(t)$ is the earliest (latest) time at which the maintenance of the machine should start (finish). If $s=t-r$, the model becomes a deterministic one, furthermore, if $s=t-r=0$, it becomes the model which referred in literature as "machine release time" (obviously the SPT rule is optimal for this case); and if $s=0$ and $s \ne t-r$, it belongs to the first kind of flexible maintenance, where there is only one maintenance activity, as discussed in Section 1.

More specifically, the problem studied in this paper, which is denoted as P, is to jointly schedule n jobs, i.e., $\{J_1, \ldots J_n\}$, and a preventive maintenance activity on a single machine. All jobs are available for processing at time zero. Both resumable and nonresumable cases are considered. Let p_i denote the processing time of job J_i and $C_i(\sigma)$ its completion time in a scheduleσ, for $i=1,\ldots,n$. The goal is to find a joint schedule of the n jobs and the maintenance to minimize the total completion time $f_P(\sigma)$, where $f_P(\sigma)=\sum_{i=1}^{n} C_i(\sigma)$. Let $[B, F]$ be the actual maintenance period with B as the starting and F the finishing time of the maintenance. It is obvious that $B \ge s$, $F \le t$ and $F-B=r$. Let $L(\sigma)$ and $R(\sigma)$ be the sets of jobs completed before and after the maintenance in schedule σ, respectively. Define $A_i=\sum_{j=1}^{i} p_j$, the total processing time of the first i jobs. Without loss of generality, let $A_0=0$. In addition, let $J_{[i]}(\sigma)$ be the ith job in schedule σ, then $p_{[i]}(\sigma)$ and $C_{[i]}(\sigma)$ are its processing time and completion time, respectively, and $A_{[i]}(\sigma)$ is the total processing time of the first i jobs in schedule σ. For simplicity, we omit "(σ)" if there is no confusion.

3. Basic properties

In this section, three simple properties of optimal solutions to each problem outlined in Section 2 are identified. In the following properties, let σ be an optimal schedule, while L and R are the sets of jobs completed before and after the maintenance in schedule σ, respectively.

We start with the nonresumable case. In this case, the first job in set R is scheduled after the maintenance even if there exists idle time δ, which is a difference between the completion time of the last job in set L and the starting time of the maintenance.

3.1. The nonresumable case

Property 1. In schedule σ for the nonresumable case, we have $B=\max\{s, A_{[|L|]}\}$, where $|L|$ denotes the number of jobs in set L.

Proof. The problem has the following two cases.

Case (i): $s \ge A_{[|L|]}$.[2]

Suppose $B>s$ in schedule σ, and there is another schedule σ', which is the same as σ except that $B'=s$, where B' is the starting time of the maintenance in schedule σ'.

Case (ii): $s < A_{[|L|]}$.

Suppose $B > A_{[|L|]}$ in schedule σ, and there is another schedule σ', which is the same as σ except that $B'=A_{[|L|]}$, where B' is the starting time of the maintenance in schedule σ'.

For both cases, it is obvious that $C_{[i]}(\sigma) > C_{[i]}(\sigma')$, for $i=|L|+1,\ldots,n$. Note that $C_{[i]}(\sigma)=C_{[i]}(\sigma')$ for $i=1,\ldots,|L|$. Then we have $f_P(\sigma) > f_P(\sigma')$, so schedule σ' is better than schedule σ, schedule σ is not optimal, a contradiction. Therefore the property holds. □

Property 2. In schedule σ for the nonresumable case, we have $A_{[|L|]}+p_{[|L|+1]} > t-r$.

Proof. Suppose $A_{[|L|]}+p_{[|L|+1]} \le t-r$ in schedule σ, and there is another schedule σ', which is the same as scheduleσ except that job $J_{[|L|+1]}$ is scheduled before the maintenance. There are three cases according to the three possible value ranges of $A_{[|L|]}+p_{[|L|+1]}$: Case (i) $A_{[|L|]}+p_{[|L|+1]} \le s$; Case (ii) $A_{[|L|]} < s$ and $s \le A_{[|L|]}+p_{[|L|+1]} \le t-r$; Case (iii) $s < A_{[|L|]} < t-r$. By Property 1, we know that schedule σ' in these three cases should be as shown in Fig. 1(a)–(c). It is easy to verify that $C_{[i]}(\sigma') < C_{[i]}(\sigma)$ for $i=|L|+1,\ldots,n$ in cases (i) and (ii), and in case (iii), $C_{[|L|+1]}(\sigma') < C_{[|L|+1]}(\sigma)$ and $C_{[i]}(\sigma')=C_{[i]}(\sigma)$, for $i=|L|+2,\ldots,n$. That is, the relation $f_P(\sigma') < f_P(\sigma)$ holds in all the three cases. So schedule σ is not an optimal one, a contradiction. Therefore this property holds. □

Property 3. In schedule σ for the nonresumable case, all jobs in set L follow the SPT rule, so do the jobs in set R.

Proof. This property obviously holds since the SPT rule is optimal for the conventional subproblem that contains only these jobs. □

3.2. The resumable case

In this case, the processing of the first job in set R is different from that of the nonresumable case. Since the disrupted job can resume after the maintenance, it should start as early as possible, i.e., when there is no idle time between the last job in set L and the maintenance, it should start after the maintenance; otherwise, it should start immediately after the last job in set L. In this case, it is a "cross-job" (denoted as J_c), which starts before the maintenance and finishes after it.

The properties for the resumable case are presented below. Since the proof of Property 4 and Property 5 are similar to the corresponding nonresumable case, we omit them here.

Property 4. In schedule σ for the resumable case, we have $B \in [\max\{s, A_{[|L|]}\}, t-r]$, and the objective value, i.e., the total

[2] This case is not rare in air-conditioner manufacturers due to: (i) $t-s$ is not very much larger than r as the resources required for the maintenance are limited; (ii) $t-s$ is not very much larger than the average job processing times as a batch of products with the same type will be considered as one job because they are arranged together due to the consideration of efficiency.

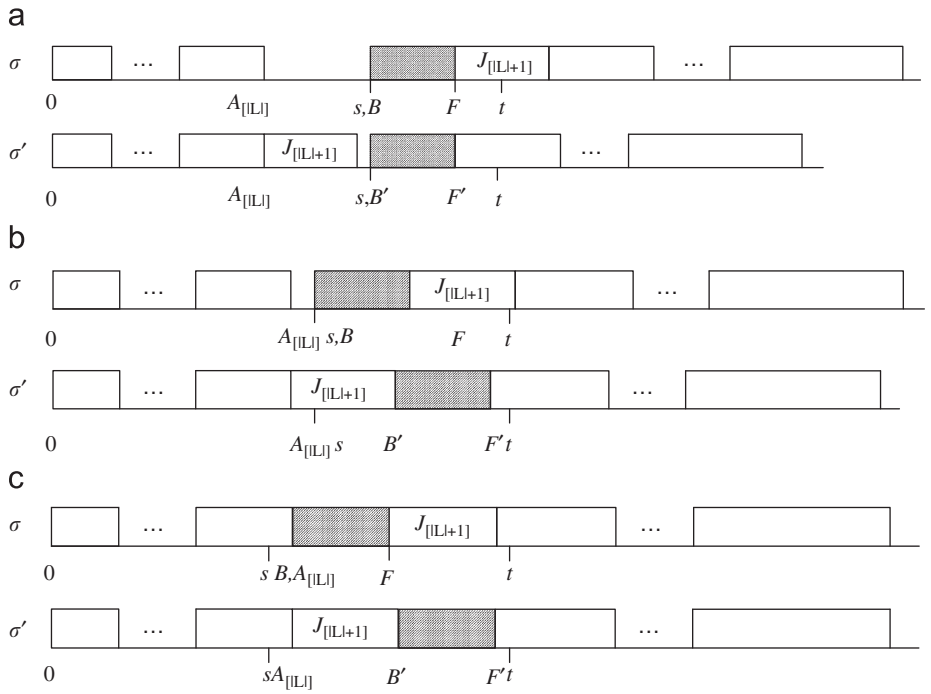

Fig. 1. (a) Illustration of Case (i) for Property 2. (b) Illustration of Case (ii) for Property 2. (c) Illustration of Case (iii) for Property 2.

completion time, keeps constant when the starting time B is in this interval.

Property 5. In schedule σ for the resumable case, we have $A_{[|L|]}+p_{[|L|+1]}>t-r$.

Property 6. In schedule σ for the resumable case, all jobs in set L follow the SPT rule, so do the jobs in set R.

Proof. It is obvious that in an optimal schedule all jobs in set L should follow SPT rule, so should the jobs in set $R\backslash\{J_c\}$. Then only the relationship between job J_c and its immediate successor needs to be considered. By using a job-swap scheme, it can be easily proved that these two jobs also follow SPT rule. The process of the proof is omitted here. □

4. SPT algorithm and its performance analysis

Based on the above properties, a SPT algorithm is proposed for both cases.

Step 1. Renumber the jobs according to the SPT rule, i.e., $p_1\le p_2\le\ldots\le p_n$. Let $L=\varnothing$, $R=\varnothing$, $l=0$ and $i=1$;

Step 2. If $i>n$, go to Step 4;

Step 3. If $l+p_i\le t-r$, schedule job J_i in set L, update L, l and i such that $L=L\cup\{J_i\}$, $l=l+p_i$ and $i=i+1$, go to Step 2; otherwise, schedule jobs J_i, J_{i+1}, …, J_n in set R, update R such that $R=R\cup\{J_i,J_{i+1},\ldots J_n\}$;

Step 4. Let s_R be the starting time of the first job in set R. If $A_{|L|}\ge s$, let $B=A_{|L|}$ and $s_R=A_{|L|}+r$; otherwise, let $B=s$ for both cases, and $s_R=A_{|L|}$ for the resumable case, $s_R=s+r$ for the nonresumable case. Stop.

The performance of the algorithm for both cases is discussed below.

Theorem 1. *For the resumable case, the SPT algorithm is optimal.*

Proof. It can be shown that the last job in set L and the first job in set R, namely $J_{[|L|]}$ and $J_{[|L|+1]}$, follow the SPT rule in an optimal schedule by using a job-swap scheme. Combing with Property 6, we can conclude that all jobs follow the SPT rule in an optimal schedule. Thus, the optimal schedules are originated from the SPT sequence. From Property 5, we know that in an optimal schedule, job J_1 to job $J_{|L|}$ should be processed before the maintenance such that $t-r-p_{|L|+1}<A_{|L|}\le t-r$, i.e., the optimal schedules have fixed sets L and R. Since jobs are resumable, there is no idle time between any two jobs or a job and the maintenance. Then the difference among these schedules is the starting time of the maintenance. From Property 4, we have $B\in[\max\{s,A_{|L|}\},t-r]$ and the objective value keeps constant when B is in this interval. In addition, we can also find that the job sets generated by the SPT algorithm are identical to the ones we discussed before, and the starting time of the maintenance generated by the algorithm is $B=\max\{s,A_{|L|}\}$. Therefore, the schedule generated by the SPT algorithm is one of the optimal schedules. □

Theorem 2. *For the nonresumable case, the SPT algorithm is optimal if it leads to* $B=A_{|L|}$.

Proof. According to Property 3, we know that in an optimal schedule, jobs scheduled before the maintenance and those after the maintenance should follow the SPT rule, respectively. Similar to the way to prove Theorem 1, if we can show that the last job scheduled before the maintenance and the first job after the maintenance follow the SPT rule under the specified condition of this theorem, then the SPT algorithm is optimal. Let L^* and R^* be the sets of jobs that are scheduled before and after the maintenance in an optimal schedule σ^*, and σ be a schedule obtained by swapping two adjacent jobs J_i and J_j in σ^* (J_i is the last job in set L^* and J_j is the first job in set R^*). After the swap, the corresponding sets of jobs in schedule σ are L and R, respectively. Assume that the jobs J_i and J_j do not follow the SPT rule, i.e., $p_i>p_j$. As J_i is the last job in set L^* and J_j is the first job in set R^*, then $A_{[|L^*|-1]}+p_i\le t-r$ and $A_{[|L^*|-1]}+p_i+p_j>t-r$. Since $p_i>p_j$, the inequality $A_{[|L^*|-1]}+p_j<t-r$ holds, that is, job J_j is scheduled before the maintenance in schedule σ. And the inequality $A_{[|L^*|-1]}+p_i+p_j>t-r$ implies that job J_i is scheduled after the maintenance. If schedule σ satisfy the specified condition, i.e., $B=A_{[|L|]}\ge s$, then $A_{[|L|-1]}+p_j=A_{[|L^*|-1]}+p_j\ge s$. Therefore, $A_{[|L^*|-1]}+p_i\ge s$. This means there is no idle time in schedule σ^*

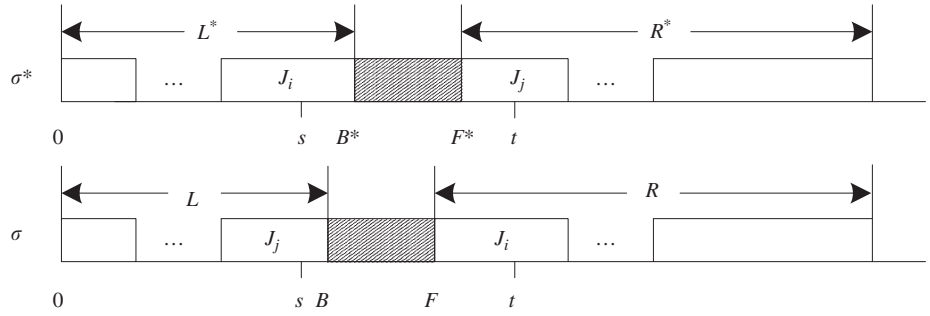

Fig. 2. Illustration of Theorem 2.

either, as shown in Fig. 2. Then the relationships of the completion times of the two jobs in schedules σ and σ^* are $C_j(\sigma)<C_i(\sigma^*)$ and $C_i(\sigma)=C_j(\sigma^*)$, respectively. Hence $C_j(\sigma)+C_i(\sigma)<C_i(\sigma^*)+C_j(\sigma^*)$. For the jobs other than J_i and J_j, the completion times in the two schedules are equal. Then $f_P(\sigma)<f_P(\sigma^*)$, so schedule σ^* is not optimal, a contradiction. The proof is completed. □

Theorem 3. *For the nonresumable case, the SPT algorithm is optimal if it leads to* $|R|\le 1$ *or* $|R|=n$.

Proof. It is obvious that the SPT algorithm is optimal if $|R|=0$ or $|R|=n$, since when $|R|=0$ or $|R|=n$, the problem reduced to the classical unconstrained one or the one with machine release time. If $|R|=1$, from Theorem 2, we know that only the case with $\delta>0$ need to be considered. By a job-swap scheme, we can get that a schedule in which the last job in set L and the first job in set R, namely $J_{[n-1]}$ and $J_{[n]}$, follow the SPT rule is not worse than the corresponding one in which these two jobs do not follow the SPT rule. By Property 3, we can conclude that the best schedules originated from the SPT sequence are not worse than those from other sequences. In addition, Property 2 indicates that the best schedules have the same sets of L and R as those generated by the SPT algorithm. The difference among the schedules with the fixed sets L and R is the starting time of the maintenance. From Property 1, we know that in an optimal schedule,$B=\max\{s,A_{|L|}\}$, which is the same as the corresponding starting time of the maintenance generated by the SPT algorithm. Thus, the SPT algorithm is optimal.

Theorems 2 and 3 are proposed as a basis for the branch and bound introduced in Section 6. For the completeness of the theory, Theorem 4 is also proposed as follows, which shows that relaxing the starting time of the maintenance cannot improve the relative error bound of the SPT algorithm. □

Theorem 4. *For the nonresumable case, the SPT algorithm has a tight worst-case error bound of* 2/7.

Proof. From Theorems 2 and 3, we know that only the case with $\delta>0$ and $1<|R|<n$ need to be considered. According to whether there exists idle time or not in the optimal schedule, we consider two cases: when idle time exists, it is exactly the same case in Lee and Liman [6]; otherwise, the proof is similar to that proposed in Lee and Liman [6], which is elaborated in Appendix A.

The following example is proposed to show the tightness of this worst-case error bound, in which, $p_1=1$, $p_2=\mu$, $p_3=\mu$, $p_4=\mu$, $r=1$, $s=\mu-1$, and $t=\mu+1$. The SPT algorithm will schedule the jobs in the sequence of $J_1-J_2-J_3-J_4$ with $B=s=\mu-1$ and $F=\mu$, then the objective value is $f_P(SPT)=9\mu+1$. One of the possible optimal schedules is $J_2-J_1-J_3-J_4$ with $B=\mu$ and $F=t=\mu+1$, and $f_P^*=7\mu+6$. Therefore, the relative worst-case error bound is $R_{SPT}=(f_P(SPT)-f_P^*)/f_P^*=(2\mu-5)/(7\mu+6)$. Hence R_{SPT} tends to 2/7 when μ tends to infinity. The proof is completed. □

5. Dynamic programming

According to Property 3, we know that the jobs scheduled before the maintenance should follow the SPT rule, so should the jobs scheduled after the maintenance. Therefore, we can use dynamic programming algorithm by renumbering all jobs in the order of SPT rule firstly and then scheduling every job before or after the maintenance one by one.

Now we suppose that all jobs have been in the SPT order and we have scheduled jobs from job J_1 to job J_i, the minimum total completion time is $f^B(i,l)$, where the maintenance starts at time B, and the total processing time of jobs scheduled before the maintenance is l. Then the initial condition ($i=1$) and the recursive equation ($i\ge 2$) are calculated as follows:

Initial condition:

$$f^B(1,l)=\begin{cases} p_1, & \text{if } l=p_1\\ B+r+p_1, & \text{if } l=0\\ \infty, & \text{otherwise}\end{cases} \tag{1}$$

for each possible value of B.

Recursive equation:

$$f^B(i,l)=\begin{cases} \min\{f^B(i-1,l-p_i)+l, f^B(i-1,l)+B+r+A_i-l\}, & \text{if } l-p_i\ge 0\\ f^B(i-1,l)+B+r+A_i-l, & \text{otherwise}\end{cases} \tag{2}$$

for each possible value of B, where $i=2,\ldots,\ n,\ B=s,\ldots,t-r$, $l=0,\ldots,t-r$.

Optimal objective value: $f_P^*=\min\{f^B(n,l): B=s,\ldots,t-r,\ l=0,\ldots,t-r\}$.

Time complexity: In order to obtain the optimal solution, we need to compute $f^B(i,l)$ for all values of i, B, and l. It is obvious that there are n possible values of i, $(t-r-s+1)$ possible values of B and $(t-r+1)$ possible values of l. Therefore, the total number of arithmetical operations is $O(n(t-r+1)(t-r-s+1))$, i.e., the time complexity of this algorithm is $O(n(t-r+1)(t-r-s+1))$. If $t-r=s$, the time complexity is $O(n(s+1))$, which is consistent with the one for the deterministic maintenance problem in Kacem et al. [14].

Memory requirement: As we know from above, for each value of i, we need to compute $(t-r+1)(t-r-s+1)$ possible values of $f^B(i,l)$s from the same quantity of possible values of $f^B(i-1,l)$s, and save them into the computer memory. Then we release the memory for the $f^B(i-1,l)$s. So the memory requirement of this algorithm is$O((t-r+1)(t-r-s+1))$.

Justification of Eq. (2): Any job J_i can be scheduled either before or after the maintenance as long as it can result in a smaller total completion time. If job J_i is scheduled before the maintenance,

then the total completion time is $f^B(i-1,l-p_i)+l$, and otherwise it is $f^B(i-1,l)+B+r+A_i-l$.

6. Branch-and-bound

In this section, we propose a branch-and-bound algorithm to solve the nonresumable problem optimally. It is assumed that the jobs are renumbered in the SPT order, furthermore, we assume that $\delta>0$ in the SPT schedule otherwise it is optimal.

6.1. Branch-and-bound schema

The proposed algorithm uses an usual search tree: a level i represents the number of scheduled jobs, a vector $x=[x_1,x_2,\ldots,x_i]$ defines a partial solution, where $x_i\in\{0,1\}$, and $x_i=1$ ($x_i=0$) corresponds to the left (right) child node, implying that job J_i is scheduled before (after) the maintenance, i.e., in set $L(R)$.

For a given node in level i, whose partial vector is $x=[x_1,x_2,\ldots,x_i]$, define $A_L^i=\sum_{j=1}^{i}x_jp_j$, $f_L^i=\sum_{j=1}^{i}\sum_{k=1}^{j}x_kp_k$ and $f_R^i=(B+r)\sum_{j=1}^{i}(1-x_k)+\sum_{j=1}^{i}\sum_{k=1}^{j}(1-x_k)p_k$, that is, the completion time of the last job in set L is A_L^i, the total completion time of the jobs in set L and set R are f_L^i and f_R^i, respectively. If we consider an $n-i$ jobs (from J_{i+1}up to J_n) problem P', in which $s'=s-A_L^i$, $t'=t+A_i-2A_L^i$, $r'=r+A_i-A_L^i$, $B'=B-A_L^i$ and $F'=B+r+A_i-2A_L^i$, as shown in Fig. 3, then we have $f_P=f_L^i+f_R^i+(n-i)A_L^i+f_{P'}$, where $f_{P'}$ is the objective value of the problem P'. It is obvious that f_R^i and $f_{P'}$ are all functions of the starting time B.

If one of the following conditions is satisfied, then an optimal solution for this node is obtained. Therefore we can update the current best objective value if the value in such a solution is smaller and then remove this node from the node list.

(1) If $p_{i+1}>t-r-A_L^i$, then schedule all the remaining jobs after the maintenance one by one, and let $B=\max\{A_L^i,s\}$.
(2) If $A_n-A_i\le s-A_L^i$, then schedule all the remaining jobs before the maintenance one by one, and let $B=s$.

Otherwise, we compute its lower bound $LB=f_L^i+(n-i)A_L^i+\min_{\max\{s,A_L^i\}\le B\le t-r}\{f_R^i+LB(P')\}$, where $LB(P')$ is the lower bound for the remaining jobs under a given starting time B, i.e., it is a lower bound for the problem with fixed maintenance. For the fixed time B, if the SPT rule can lead to one of the following two conditions: (1) $\delta'=0$ (2) $|L'|=0$ or $|L'|=n-i-1$ or $|L'|=n-i$, where L' is the set of jobs scheduled before the maintenance and δ' is the idle time, then it is optimal (this conclusion can be deduced from Theorems 2 and 3 by setting $s=t-r$). Thus we can obtain a complete solution and can further update the current best objective value if the value in such a solution is smaller. In this case, this optimal objective value of problem P' can be considered as the lower bound under this fixed starting time B. Otherwise, after a transformation, the lower bound in Section 6.3 can be considered as our lower bound $LB(P')$. Then we compare the lower bound LB with the current best objective value, if such a bound is smaller, insert it in the node list such that the nodes in the list are in an increasing order according to their lower bounds, otherwise eliminate it.

6.2. Initial solution

At first, we propose two properties which can be used in a heuristic H to obtain the initial solution. Denote σ as the schedule obtained by the SPT algorithm. Then we try to exchange the last job scheduled before the maintenance with one of the jobs scheduled after the maintenance. During this process the jobs are reordered before and after the maintenance according to the SPT rule once again.

Property 7. In scheduleσ, if there exists two jobs J_i and J_j in set R such that $p_i\le p_{|L|}+\delta, p_j\le p_{|L|}+\delta$ and $p_{|L|}\le p_i\le p_j$, then $f_P(\sigma_2)\le f_P(\sigma_1)\le f_P(\sigma)$, where σ_1 and σ_2 are the schedule obtained by swapping job $J_{|L|}$ for J_i, job $J_{|L|}$ for J_j, respectively.

Proof. It is easy to obtain this property by computing $C_{[k]}(\sigma)$, $C_{[k]}(\sigma_1)$ and $C_{[k]}(\sigma_2)$, for $\forall|L|\le k\le j$, as shown in Fig. 4. □

Property 8. In schedule σ, if there exists two jobs J_i and J_j in set R such that $p_{|L|}+\delta<p_i\le p_{|L|}+\delta+t-r-s$, $p_{|L|}+\delta<p_j\le p_{|L|}+\delta+t-r-s$ and $p_i\le p_j$, then $f_P(\sigma_2)\ge f_P(\sigma_1)$, where σ_1 and σ_2 are the schedule obtained by swapping job $J_{|L|}$ for J_i, job $J_{|L|}$ for J_j, respectively.

Proof. It is easy to obtain this property by computing $C_{[k]}(\sigma)$, $C_{[k]}(\sigma_1)$, and $C_{[k]}(\sigma_2)$, for $\forall|L|\le k\le j$, as shown in Fig. 5. □

Based on these, we present the heuristic H as follows.

Step 1. Schedule the jobs on the machine according to the SPT algorithm in Section 4. Notations L, R, and δ are defined as before. Then an objective value f_1 can be obtained.

Step 2. Find the biggest j ($|L|<j\le n$) such that $p_j\le p_{|L|}+\delta$, if it exists, schedule job J_j as the last job in set L and $J_{|L|}$ as the first job

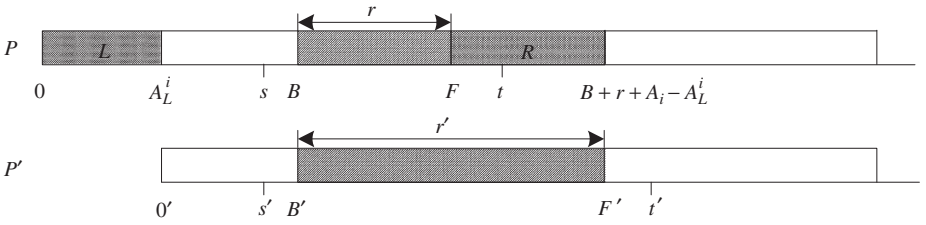

Fig. 3. Illustration of the lower bound transformation.

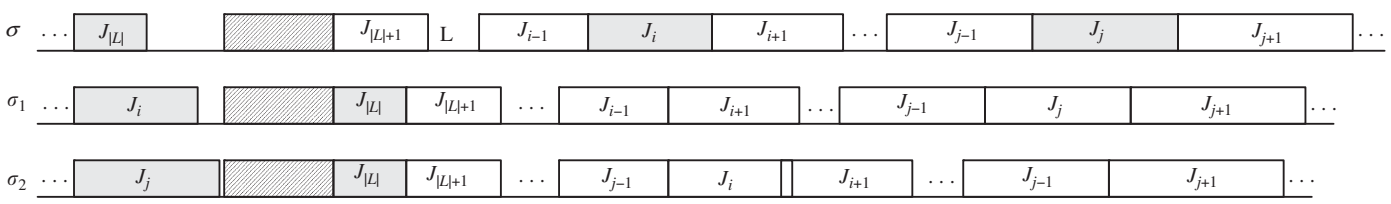

Fig. 4. Illustration of Property 7.

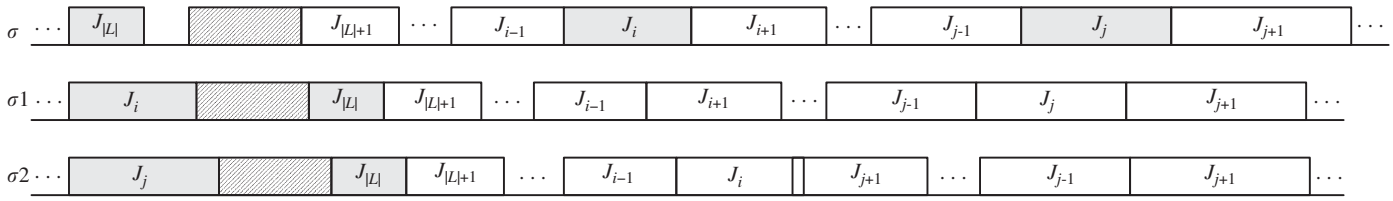

Fig. 5. Illustration of Property 8.

in set R, then an objective value f_2 can be obtained, and in its corresponding schedule we have $B=s$.

Step 3. Find the smallest j ($|L|<j\le n$) such that $p_{|L|}+\delta< p_j\le p_{|L|}+\delta+t-r-s$, if it exists, schedule job J_j as the last job in set L and $J_{|L|}$ as the first job in set R, then an objective value f_3 can be obtained, and in its corresponding schedule we have $B=A_{|L|-1}+p_j$.

Step 4. Let the initial objective value be$f_P=\min\{f_1,f_2,f_3\}$.

6.3. Lower bounds for the problem with a fixed starting time of the maintenance

In this section, we propose several lower bounds based on different perspectives, and then compare them analytically; the lower bounds which are not dominated by others will be used in the branch-and-bound schema.

Denote L and R as the set of jobs scheduled before and after the maintenance by applying the SPT rule to the problem with a fixed starting time B (denoted as φ), respectively.

6.3.1. Lower bound based on flexible maintenance

Proposition 1. $LB_1=\sum_{i=1}^{n}A_i+(n-|L|-1)r$ is a valid lower bound.

Proof. Let φ' be a relaxed problem in which the starting time of the maintenance can be in an interval $[B,A_{|L|+1}]$. Then we have $f_\varphi=f^*_{\varphi'}$. From Theorem 2, we know that the SPT algorithm can obtain an optimal solution of this relaxed problem, with the optimal objective value of $f^*_{\varphi'}=\sum_{i=1}^{n}A_i+(n-|L|-1)r$. Therefore this conclusion holds. □

Proposition 2. $LB_2=\sum_{i=1}^{n}A_i+(n-|L|)r$ is a valid lower bound.

Proof. The conclusion is based on a different relaxed problem: the starting time of the maintenance can be in an interval $[A_{|L|},B]$. □

Remark. The lower bound LB_2 can also be obtained by relaxing the nonresumable constraint to a resumable one.

In view of the relationship between these two lower bounds, the following proposition holds obviously.

Proposition 3. $LB_2>LB_1$.

6.3.2. Lower bound based on mixed integer linear model

Proposition 4. $LB_3=\min\{\theta_1,\theta_2\}$ is a valid lower bound, where

$$\theta_1=\sum_{i=1}^{n}A_i+(n-|L|)r+\frac{1}{p_{|L|}}(r+p_{|L|+1})(A_{|L|-1}+p_{|L|+1}-B) \tag{3}$$

and

$$\theta_2=\sum_{i=1}^{n}A_i+(n-|L|)r+\frac{1}{p_{|L|+2}}(r+p_{|L|+1}-p_{|L|+2})(A_{|L|}-B) \tag{4}$$

Proof. From a lower bound proposed by Kacem and Chu [15], using a mixed integer linear model, for the weighted total completion time problem, we can easily obtain the corresponding lower bound for the total completion time problem

$$LB_3=\sum_{i=1}^{|L|+1}A_i+\sum_{i=|L|+2}^{n}(A_i+r)+\frac{r}{p_{|L|+1}}(p_{|L|+1}-\delta)+\min\left(\lambda_1^*\left(\delta+\frac{r\delta}{p_{|L|+1}}\right),\lambda_2^*(p_{|L|+1}-\delta)\left(1+\frac{\delta}{p_{|L|+1}}\right)\right) \tag{5}$$

where

$$\lambda_1^*=\begin{cases}1-\dfrac{p_{|L|+1}}{p_{|L|+2}}, & \text{if } J_{|L|+2} \text{ exists}\\ 1 & \text{otherwise}\end{cases} \tag{6}$$

and

$$\lambda_2^*=\frac{p_{|L|+1}}{p_{|L|}}-1 \tag{7}$$

From Section 6.1, we know that under the fixed starting time B, if there is only one job after the maintenance in the SPT schedule (this means job $J_{|L|+2}$ does not exist), then this schedule is optimal. So in the branch-and-bound schema, one of the conditions under which we need to compute the lower bound is that job $J_{|L|+2}$exists. Therefore Eq. (6) can be rewritten as

$$\lambda_1^*=1-\frac{p_{|L|+1}}{p_{|L|+2}} \tag{8}$$

By substituting Eqs. (7) and (8) into Eq. (5) we can obtain this lower bound. □

6.3.3. Lower bound based on job splitting

The idea of job splitting was proposed by Posner [48] to produce a lower bound for the problem of scheduling jobs with deadlines on a single machine to minimize total weighted completion time. Then it was used by Belouadah et al. [49] and Webster [50] to solve total weighted completion time problems in other different cases (the job splitting method is explained in general by Belouadah [49]). Kacem et al. [14] and Kacem and Chu [15] also used this method to build lower bounds for the single machine total weighted completion time problem with a fixed maintenance.

As defined previously, φ denote the original problem with no split jobs. Define φ' as the corresponding problem in which job J_i is split into n_i pieces, where each piece (J_i,k) have a processing time p_i^k and a weight w_i^k ($\forall 1\le k\le n_i$) such that $\sum_{k=1}^{n_i}p_i^k=p_i$ and $\sum_{k=1}^{n_i}w_i^k=w_i$. Then Belouadah et al. [49] established the following relation:

$$w_iC_i=\sum_{k=1}^{n_i}w_i^kC_i^k+\sum_{k=1}^{n_i-1}w_i^k\Big(\sum_{l=k+1}^{n_i}p_i^l\Big) \tag{9}$$

where C_i is the completion time of job J_i in a schedule for problem φ and C_i^k is the completion time of piece (J_i,k) in a schedule for problem φ' in which pieces are scheduled contiguously. Based on

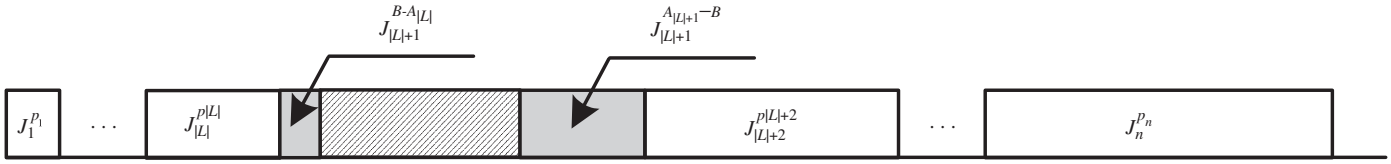

Fig. 6. Illustration of Proposition 5.

this, Kacem et al. [14] proposed the following relation:

$$f_{\varphi}^{*} \geq f_{\varphi''}^{*} + \sum_{i=1}^{n}\left(\sum_{k=1}^{n_i-1} w_i^k\left(\sum_{l=k+1}^{n_i} p_i^l\right)\right) \tag{10}$$

where φ'' is a problem obtained by relaxing the contiguity constraint of problem φ'.

The job splitting technique is extremely useful in solving the total weighted completion time problems. To apply this method, we assume $w_i=1$ for all $i=1,\ldots,n$. In this section, we propose two valid lower bounds based on different kinds of split.

Proposition 5. $LB_4=\sum_{i=1}^{n}A_i+(n-|L|+(A_{|L|}-B)/p_{|L|+1})r$ is a valid lower bound.

Proof. Consider the following split: $\forall 1\leq i\leq n$, job J_i is split into p_i pieces such that $p_i^k=1$ and $w_i^k=1/p_i$. Then problem φ'' is to schedule these A_n pieces on the single machine. As shown in Fig. 6, where J_i^k denotes the set of k pieces of job J_i, the WSPT rule can solve it optimally with the objective value of

$$\begin{aligned}
LB_4 &= \sum_{i=1}^{|L|}\left(\frac{1}{p_i}\left(p_iA_{i-1}+\frac{(1+p_i)p_i}{2}\right)+\frac{1}{p_i}\sum_{k=1}^{p_i-1}\sum_{l=k+1}^{p_i}1\right)\\
&\quad+\frac{1}{p_{|L|+1}}\left((B-A_{|L|})A_{|L|}+\frac{(1+B-A_{|L|})(B-A_{|L|})}{2}\right)\\
&\quad+\frac{1}{p_{|L|+1}}\left((A_{|L|+1}-B)(B+r)+\frac{(1+A_{|L|+1}-B)(A_{|L|+1}-B)}{2}\right)\\
&\quad+\frac{1}{p_{|L|+1}}\sum_{k=1}^{p_{|L|+1}-1}\sum_{l=k+1}^{p_{|L|+1}}1\\
&\quad+\sum_{i=|L|+2}^{n}\left(\frac{1}{p_i}\left(p_i(A_{i-1}+r)+\frac{(1+p_i)p_i}{2}\right)+\frac{1}{p_i}\sum_{k=1}^{p_i-1}\sum_{l=k+1}^{p_i}1\right)\\
&=\sum_{i=1}^{|L|}A_i+\frac{1}{p_{|L|+1}}(A_{|L|+1}p_{|L|+1}-Br+A_{|L|+1}r)+\sum_{i=|L|+2}^{n}(A_i+r)\\
&=\sum_{i=1}^{n}A_i+\left(n-|L|+\frac{A_{|L|}-B}{p_{|L|+1}}\right)r
\end{aligned}$$

Remark. This lower bound is equal to the one obtained by the following split:$\forall i\neq|L|+1$, job J_i is not split, but $J_{|L|+1}$ is split into two pieces $(J_{|L|+1},1)$ and $(J_{|L|+1},2)$ such that $p_{|L|+1}^1=B-A_{|L|}$, $w_{|L|+1}^1=(B-A_{|L|})/p_{|L|+1}$, $p_{|L|+1}^2=A_{|L|+1}-B$ and $w_{|L|+1}^2=(A_{|L|+1}-B)/p_{|L|+1}$ (see [14]).

Since $A_{|L|}<B$, the following proposition holds accordingly.

Proposition 6. $LB_4<LB_2$.

Considering another split, we can obtain the following proposition.

Proposition 7. Let J_x be the job such that $A_{x-1}\leq B-p_{|L|+1}$and $A_x>B-p_{|L|+1}$. Then $LB_5=\min\{\gamma_1,\gamma_2\}$ is a valid lower bound, where

$$\begin{aligned}
\gamma_1 &= \sum_{i=1}^{n}A_i+B-A_{|L|}+(n-x)r+(|L|-x)p_{|L|+1}\\
&\quad+\frac{1}{p_x}(r+p_{|L|+1})(A_{x-1}+p_{|L|+1}-B)
\end{aligned}\tag{11}$$

and

$$\gamma_2=\sum_{i=1}^{n}A_i+(n-|L|)r+\frac{1}{p_{|L|+2}}(r+p_{|L|+1}-p_{|L|+2})(A_{|L|}-B)\tag{12}$$

Proof. This proposition is based on the following split: $\forall i\neq|L|+1$, job J_i is split into p_i pieces such that $p_i^k=1$ and $w_i^k=1/p_i$, but $J_{|L|+1}$is not split, i.e., it remains one piece with $p_{|L|+1}$and$w_{|L|+1}=1$. Problem φ'' is to schedule these $A_n-p_{|L|+1}+1$ pieces on the single machine. It is obvious that those pieces scheduled before the maintenance should follow the WSPT rule, so should the pieces scheduled after the maintenance. Since the processing times of all pieces except for job $J_{|L|+1}$ are equal, this problem is reduced to scheduling $J_{|L|+1}$ before or after the maintenance, thus one of the two schedules in Fig. 7 must be optimal. Schedule σ_1 leads to the objective value γ_1, which is calculated as follows:

$$\begin{aligned}
\gamma_1 &= \sum_{i=1}^{x-1}\left(\frac{1}{p_i}\left(p_iA_{i-1}+\frac{(1+p_i)p_i}{2}\right)+\frac{1}{p_i}\sum_{k=1}^{p_i-1}\sum_{l=k+1}^{p_i}1\right)+B\\
&\quad+\sum_{i=x+1}^{|L|}\left(\frac{1}{p_i}\left(p_i(A_{i-1}+r+p_{|L|+1})+\frac{(1+p_i)p_i}{2}\right)+\frac{1}{p_i}\sum_{k=1}^{p_i-1}\sum_{l=k+1}^{p_i}1\right)\\
&\quad+\frac{1}{p_x}\left((B-A_{x-1}-p_{|L|+1})A_{x-1}+\frac{(1+B-A_{x-1}-p_{|L|+1})(B-A_{x-1}-p_{|L|+1})}{2}\right)\\
&\quad+\frac{1}{p_x}\left((p_x+A_{x-1}+p_{|L|+1}-B)(B+r)\right.\\
&\quad\left.+\frac{(1+p_x+A_{x-1}+p_{|L|+1}-B)(p_x+A_{x-1}+p_{|L|+1}-B)}{2}\right)\\
&\quad+\frac{1}{p_x}\sum_{k=1}^{p_x-1}\sum_{l=k+1}^{p_x}1+\sum_{i=|L|+2}^{n}\left(\frac{1}{p_i}\left(p_i(A_{i-1}+r)+\frac{(1+p_i)p_i}{2}\right)+\frac{1}{p_i}\sum_{k=1}^{p_i-1}\sum_{l=k+1}^{p_i}1\right)\\
&=\sum_{i=1}^{x-1}A_i+B+\sum_{i=x+1}^{|L|}A_i+(|L|-x)(r+p_{|L|+1})\\
&\quad+\frac{1}{p_x}(A_{x-1}+p_{|L|+1}-B)(r+p_{|L|+1}+p_x)\\
&\quad+B+r+\frac{1+p_x}{2}+\frac{p_x-1}{2}+\sum_{i=|L|+2}^{n}A_i+(n-|L|-1)r\\
&=\sum_{i=1}^{n}A_i+B-A_{|L|}+(n-x)r+(|L|-x)p_{|L|+1}+\frac{1}{p_x}(r+p_{|L|+1})(A_{x-1}+p_{|L|+1}-B)
\end{aligned}$$

and the other schedule σ_2 results in the objective value γ_2, which is calculated as follows:

$$\begin{aligned}
\gamma_2 &= \sum_{i=1}^{|L|}\left(\frac{1}{p_i}\left(p_iA_{i-1}+\frac{(1+p_i)p_i}{2}\right)+\frac{1}{p_i}\sum_{k=1}^{p_i-1}\sum_{l=k+1}^{p_i}1\right)\\
&\quad+B+r+p_{|L|+1}+\frac{1}{p_{|L|+2}}\left((B-A_{|L|})A_{|L|}+\frac{(1+B-A_{|L|})(B-A_{|L|})}{2}\right)\\
&\quad+\frac{1}{p_{|L|+2}}\left((p_{|L|+2}+A_{|L|}-B)(B+r+p_{|L|+1})\right.\\
&\quad\left.+\frac{(1+p_{|L|+2}+A_{|L|}-B)(p_{|L|+2}+A_{|L|}-B)}{2}\right)+\frac{1}{p_{|L|+2}}\sum_{k=1}^{p_{|L|+2}-1}\sum_{l=k+1}^{p_{|L|+2}}1\\
&\quad+\sum_{i=|L|+3}^{n}\left(\frac{1}{p_i}\left(p_i(A_{i-1}+r)+\frac{(1+p_i)p_i}{2}\right)+\frac{1}{p_i}\sum_{k=1}^{p_i-1}\sum_{l=k+1}^{p_i}1\right)
\end{aligned}$$

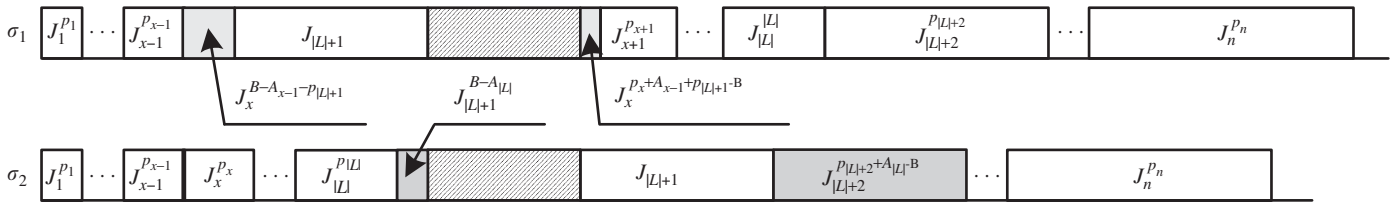

Fig. 7. Illustration of Proposition 7.

$$= \sum_{i=1}^{|L|} A_i + B + r + p_{|L|+1} + \frac{1}{p_{|L|+2}}(A_{|L|} - B)(r + p_{|L|+1}) + A_{|L|} + r + p_{|L|+1} + \frac{1+p_{|L|+2}}{2} + \frac{p_{|L|+2}-1}{2} + \sum_{i=|L|+3}^{n} A_i + (n-|L|-2)r$$

$$= \sum_{i=1}^{n} A_i + (n-|L|)r + \frac{1}{p_{|L|+2}}(r + p_{|L|+1} - p_{|L|+2})(A_{|L|} - B)$$

Proposition 8. $LB_5 \geq LB_3$.

The proposition is proved in Appendix B.

Proposition 9. $LB_5 \geq LB_4$.

The proposition is proved in Appendix C.

6.3.4. Lower bound based on SPT rule

Proposition 10. $LB_6 = \sum_{i=1}^{n} A_i + (n-|L|)r + B - A_{|L|}$ is a valid lower bound.

Proof. From the proof process of the error bound of SPT algorithm in Lee and Liman [6], we know that $f_\varphi^* \geq f_\varphi(SPT) - (|R|-1)(\delta - \delta^*)$, where δ denotes the idle time in the SPT schedule and δ^* denotes the idle time in the optimal schedule. Since $|R| = n - |L|$, $\delta = B - A_{|L|}$ and $\delta^* \geq 0$, we have $f_\varphi^* \geq_n f_\varphi(SPT) - (|R| - 1)\delta = \sum_{i=1}^{n} A_i + (n-|L|)(\delta + r) - (n-|L|-1)\delta = \sum_{i=1}^{n} A_i + (n-|L|)r + B - A_{|L|}$. □

Remark. We can also get the following relation from the proof process of the error bound of SPT algorithm:

$$\frac{f_\varphi(SPT) - f_\varphi^*}{f_\varphi^*} \leq \frac{2(|R|-1)}{|R|(|R|+1)+2} \leq \frac{2}{7} \tag{13}$$

which further leads to two other lower bounds

$$LB_7 = \frac{7}{9} f_\varphi(SPT) \tag{14}$$

and

$$LB_8 = \frac{|R|(|R|+1)+2}{|R|(|R|+3)} f_\varphi(SPT) \tag{15}$$

It is easy to show that LB_7 is dominated by LB_8 since the former is the minimum of the latter (when $|R| = 3$), and the latter is dominated by LB_6. The detailed proof is stated in Appendix D.

Proposition 11. $LB_6 > LB_2$ and $LB_6 > LB_5$.

Proof. It is obvious that the first relation holds. Since $A_{|L|} \leq B$, then

$$LB_6 \geq LB_6 + \frac{1}{p_{|L|+2}}(r + p_{|L|+1})(A_{|L|} - B) = \gamma_2 \geq \min\{\gamma_1, \gamma_2\} = LB_5.$$

In conclusion, LB_6 dominates the others, so in this paper, we use it as the lower bound. □

7. Experimental results

Experiments are carried out to evaluate the performance of the dynamic programming and the branch-and-bound algorithm. The algorithms are coded in C language and run on a Lenovo PC in the Windows Vista environment.

7.1. Data generation

A complete instance is composed of job parameters and maintenance parameters. The job parameters consist of the following two parameters:

- n: the number of jobs, which ranges from 5 to 1500;
- p_i: the processing times, which are randomly generated from an integer uniform distribution $U[1,100]$.

For each value of n, 10 sets of data are generated.

The maintenance parameters consist of the following three parameters:

- s: the earliest time at which the maintenance should start;
- r: the duration of the maintenance;
- t: the latest time at which the maintenance should finish.

From the discussion in Section 5, we know that the time complexity and memory requirement of the dynamic programming algorithm are related to $t-r$ and $t-s-r$. Furthermore, according to the discussion in Section 6.1, to get the lower bound of a node, we must compute the lower bound of the fixed maintenance problem $t-r-s$ times. So the value of $t-r$ and $t-r-s$ have impacts on the performance of these two algorithms. To evaluate these impacts, values of s, r and t are generated as follows:

- $r = \lfloor A_n/n \rfloor$;
- Two parameters α and β are defined such that $s = \lfloor \alpha A_n \rfloor$ and $t = s + r + \lfloor \beta r \rfloor$, and each of them can take one of the three values: 0.25, 0.5, 0.75.

Different combinations of α and β can lead to 9 sets of s and t, thus, for each value of n, there are 90 instances.

Note. In real production, $t-s$ is probably not very much larger than r due to the resources required for the maintenance are limited, so only small values of β are considered here.

7.2. Results

For the dynamic programming algorithm, we provide three items: the average optimal objective value of 10 generated instances $\overline{DP} = (\sum_{k=1}^{10} DP^k)/10$, the average computation time required to find the optimal solution $\bar{t}_{DP} = (\sum_{k=1}^{10} t_{DP}^k)/10$ (in seconds), the maximal computation time $t_{DP,m} = \max_{1 \leq k \leq 10} \{t_{DP}^k\}$, where DP^k and t_{DP}^k denote the optimal objective value and the computation time of the kth instance, respectively. When an

Table 1
Results of the dynamic programming algorithm.

α	n	β								
		0.25			0.5			0.75		
		$\overline{DP}$	$\overline{t}_{DP}$	$t_{DP,m}$	$\overline{DP}$	$\overline{t}_{DP}$	$t_{DP,m}$	$\overline{DP}$	$\overline{t}_{DP}$	$t_{DP,m}$
0.25	5	819.4	0	0	800.6	0	0	760.8	0	0
	10	2275.8	0	0	2235.5	0.002	0.016	2221.0	0	0
	15	4277.1	0	0	4254.6	0	0	4239.2	0	0
	20	7339.7	0	0	7277.3	0	0	7262.8	0.002	0.015
	30	16312.1	0	0	16293.3	0.008	0.015	16282.7	0.005	0.016
	40	29480.0	0.003	0.015	29480.0	0.012	0.031	29453.8	0.011	0.031
	50	45978.7	0.012	0.015	45942.3	0.017	0.031	45907.2	0.017	0.047
	60	65209.2	0.016	0.031	65145.3	0.013	0.047	65120.0	0	0
	80	112267.1	0.029	0.047	112228.8	0.042	0.078	112196.8	0.033	0.109
	100	173851.4	0.053	0.062	173718.2	0.039	0.109	173691.1	0.028	0.156
	125	256912.6	0.040	0.093	256895.1	0.062	0.172	256887.3	0.064	0.234
	150	405236.9	0.128	0.203	405141.1	0.162	0.374	405114.4	0.162	0.546
	200	694609.1	0.150	0.234	694534.3	0.170	0.468	694502.0	0.125	0.702
	300	1541274.9	0.431	0.515	1541121.4	0.651	0.983	1541050.0	0.600	1.918
	400	2752485.5	0.674	0.936	2752352.0	0.802	1.653	2752306.2	0.718	2.511
	500	4207419.6	0.921	1.435	4207340.0	1.285	2.761	4207245.0	0.351	3.510
	600	6045435.1	1.285	1.919	6045322.7	1.805	3.822	*	*	*
	700	8186709.7	2.009	2.605	8186609.2	2.753	5.163	*	*	*
	800	10686104.2	3.267	3.369	*	*	*	*	*	*
	1500	38267699.4	6.965	11.824	*	*	*	*	*	*
0.5	5	755.2	0	0	755.2	0	0	746.6	0	0
	10	2200.8	0	0	2121.7	0	0	2121.7	0	0
	15	4188.3	0.002	0.015	4154.7	0.003	0.015	4146.9	0.006	0.016
	20	7137.6	0	0	7125.3	0.012	0.016	7106.8	0.008	0.016
	30	16082.2	0.008	0.016	16036.3	0.014	0.031	16017.8	0.014	0.032
	40	29120.9	0.010	0.016	29093.2	0.017	0.047	29054.4	0.016	0.063
	50	45519.3	0.022	0.032	45461.5	0.038	0.063	45400.6	0.030	0.078
	60	64699.4	0.043	0.062	64523.4	0.039	0.125	64523.4	0.056	0.172
	80	111415.9	0.056	0.094	111415.9	0.103	0.141	111397.5	0.132	0.218
	100	172699.1	0.061	0.11	172699.1	0.119	0.219	172699.1	0.175	0.327
	125	255674.4	0.075	0.171	255674.4	0.144	0.327	255646.5	0.173	0.468
	150	403757.5	0.198	0.281	403566.4	0.192	0.546	403527.0	0.215	0.811
	200	692774.9	0.390	0.483	692572.5	0.435	0.951	692439.2	0.240	1.248
	300	1538212.0	0.591	1.060	1538143.6	1.117	2.043	*	*	*
	400	2748372.8	1.366	1.887	*	*	*	*	*	*
	700	8179845.1	4.057	5.257	*	*	*	*	*	*
0.75	5	686.2	0	0	686.2	0	0	686.2	0	0
	10	2149.6	0.003	0.015	2108.3	0.005	0.016	2071.8	0.002	0.015
	15	4085.8	0.002	0.016	4012.6	0.003	0.015	4012.6	0.002	0.016
	20	7009.2	0.005	0.016	6993.0	0.009	0.016	6978.6	0.017	0.032
	30	15857.5	0.010	0.016	15830.2	0.019	0.032	15744.2	0.016	0.047
	40	28867.7	0.026	0.031	28806.1	0.037	0.062	28806.1	0.053	0.094
	50	45132.3	0.037	0.062	45064.2	0.056	0.094	45040.4	0.070	0.140
	60	64216.8	0.051	0.094	64157.1	0.083	0.141	64113.9	0.105	0.188
	80	110970.7	0.088	0.140	110872.4	0.122	0.234	110743.7	0.086	0.328
	100	172083.2	0.142	0.188	172027.3	0.238	0.327	171943.3	0.276	0.499
	125	254824.0	0.186	0.265	254824.0	0.357	0.515	254824.0	0.533	0.780
	150	402498.0	0.324	0.546	402345.3	0.452	0.827	402345.3	0.710	1.575
	200	691020.0	0.468	0.733	691020.0	0.920	1.450	*	*	*
	300	1535689.0	0.736	1.591	*	*	*	*	*	*
	400	2745349.2	2.305	2.901	*	*	*	*	*	*

optimal solution is not found for some of the 10 instances, a symbol "*" is used to denote the entry, as shown in Table 1.

For the branch-and-bound algorithm, in addition to three items similar to those of the dynamic programming, i.e., the average objective value $\overline{BAB}=(\sum_{k=1}^{10} BAB^k)/10$, the average computation time $\overline{t}_{BAB}=(\sum_{k=1}^{10} t_{BAB}^k)/10$ and the maximal computation time $t_{BAB,m}=\max_{1\le k\le 10}\{t_{BAB}^k\}$, the following two items are also used: the average number of nodes generated to find the solution $\overline{ND}=(\sum_{k=1}^{10} ND^k)/10$, the maximal number of nodes $ND_{\max}=\max_{1\le k\le 10}\{ND^k\}$, where BAB^k, ${t_{BAB}}^k$, and ND^k denote the objective value, the computation time and the number of nodes generated for the kth instance, respectively. The computation time limit is set to 1800 s, i.e., the program is stopped if it cannot find an optimal solution within the time limit and then a current best complete solution is accepted. The same as in the dynamic programming evaluation, the symbol "*"is used to mark the entry even if among 10 instances, there is only one instance for which an optimal solution cannot be found within the time limit. To show the performance of the lower bound and the initial solution, we give the average lower bound at the root of the tree $\overline{LB}=(\sum_{k=1}^{10} LB^k)/10$ and the average initial objective value $\overline{H}=(\sum_{k=1}^{10} H^k)/10$, where LB^k denotes the lower bound and H^k the objective value by applying the heuristic H to the kth instance, respectively , as shown in Table 2.

Table 2
Results of the branch-and-bound algorithm.

α	β	n	$\overline{BAB}$	$\overline{t}_{BAB}$	$t_{BAB,m}$	$\overline{ND}$	$ND_{\max}$	$\overline{LB}$	$\overline{H}$
0.25	0.25	5	819.4	0	0	3.1	5	795.7	819.4
		10	2275.8	0	0	8.2	16	2264.8	2281.5
		15	4277.1	0	0	19.7	78	4264.6	4295.2
		20	7339.7	0	0	59.4	166	7310.3	7360.6
		30	16312.1	0.002	0.015	76.0	605	16294.2	16333.6
		40	29480.0	0	0	93.5	361	29464.6	29500.9
		50	45978.7	0.036	0.312	1753.9	15153	45941.1	46056.8
		60	65209.2	0.067	0.436	1858.4	11352	65172.5	65283.5
		80	112267.1	0.119	0.733	3973.1	24151	112229.1	112329.3
		100	172881.5	110.809	1069.848	158133.2	1172986	172792.1	173080.7
		125	256912.6	0.182	1.731	3970.3	37344	256898.6	256942.7
		150	405377.2*	401.123	1800.006	245378.4	910381	405156.8	405487.3
		200	694750.6*	720.180	1800.010	361407.3	1119259	694533.9	694816.8
		300	1542092.0*	1260.205	1800.014	695906.1	1122168	1541101.9	1542097.8
		400	2753255.1*	1080.157	1800.012	671965.2	1223098	2752336.9	2753317.5
	0.5	5	800.6	0	0	2.8	5	789.1	800.6
		10	2235.5	0	0	3.0	12	2233.0	2236.8
		15	4254.6	0	0	9.9	69	4248.8	4262.1
		20	7277.3	0	0	7.3	43	7272.7	7280.5
		30	16293.3	0	0	15.5	58	16285.3	16298.9
		40	29480.0	0.005	0.015	93.5	361	29464.6	29500.9
		50	45942.3	0.005	0.031	182.4	1009	45925.3	45977.3
		60	65145.3	0.003	0.015	45. 6	157	65139.9	65151.0
		80	112228.8	0.023	0.140	419.8	2921	112213.6	112250.0
		100	172755.9	0.050	0.281	790.9	4281	172743.6	172772.1
		125	256895.1	0.017	0.140	235.9	1840	256891.1	256900.7
		150	405141.1	0.684	6.447	6172.0	56994	405124.4	405187.1
		200	694535.4*	180.320	1800.006	114909.9	1124727	694511.0	694602.8
		300	1541258.3*	540.383	1800.007	279548.3	992057	1541067.4	1541264.1
		400	2752448.0*	540.302	1800.015	316709.7	1156332	2752311	2752510.4
	0.75	5	760.8	0	0	1.3	4	755.0	760.8
		10	2221.0	0	0	1.3	8	2220.5	2222.2
		15	4239.2	0	0	2.3	14	4237.1	4242.4
		20	7262.8	0	0	1.0	10	7262.4	7263.6
		30	16282.7	0	0	9.7	22	16277.6	16285.2
		40	29453.8	0.002	0.015	26.3	169	29449.3	29458.0
		50	45907.2	0.003	0.016	28	147	45903.6	45910.8
		60	65120.0	0	0	0	0	65120.0	65120.0
		80	112196.8	0.005	0.031	41.2	290	112193.9	112203.9
		100	172743.7	0.033	0.281	362.8	3126	172737.1	172755.1
		125	256887.3	0.006	0.031	51.9	317	256885.2	256889.0
		150	405114.4	0.025	0.109	174.6	782	405110.1	405122.2
		200	694502.0	0.083	0.733	450.8	3888	694497.2	694515.5
		300	1541050.0	0.562	3.010	2198	11379	1541045.6	1541055.8
		400	2752306.2*	180.57	1800.016	108136.7	1068636.0	2752297.2	2752337.6
0.5	0.25	5	755.2	0	0	3.6	6	746.1	755.2
		10	2200.8	0	0	23.0	49	2167.9	2215.5
		15	4188.3	0	0	33.3	73	4150.4	4210.2
		20	7137.6	0	0	64.0	194	7105.5	7164.9
		30	16082.2	0.005	0.015	321.3	1214	16021.8	16152.0
		40	29120.9	0.031	0.140	1781.2	7288	29064.5	29216.4
		50	45519.3	0.268	1.622	8543.7	43124	45431.1	45653.3
		60	64699.4	39.611	374.323	112177.2	785584	64565.4	64956.6
		80	111415.9	0.058	0.280	1959.4	9690	111375.6	111482.7
		100	171735.7	0.042	0.327	1118.8	8578	171716.0	171760.2

Table 2 (*continued*)

α	β	n	$\overline{BAB}$	$\overline{t}_{BAB}$	$t_{BAB,m}$	$\overline{ND}$	$ND_{\max}$	$\overline{LB}$	$\overline{H}$
		125	255674.4	35.477	354.676	89728.4	895011	255646.4	255740.3
		150	404357.7*	930.895	1800.011	538342.9	1002810	403561.5	404395.3
		200	693882.9*	1565.673	1800.013	924203.5	1305925	692472.0	693920.8
		300	1539253.0*	1080.007	1800.016	651408.2	1192675	1537988.0	1539253.0
		400	2749664.2*	1218.315	1800.014	784610.7	1342265	2748145.6	2749692.9
	0.5	5	755.2	0	0	3.6	6	746.1	755.2
		10	2121.7	0	0	2.5	10	2119.8	2121.8
		15	4154.7	0	0	18.8	48	4131.0	4169.5
		20	7125.3	0.002	0.015	44.6	174	7098.9	7148.9
		30	16036.3	0.002	0.015	125.0	388	16000.6	16066.0
		40	29093.2	0.031	0.188	874.3	5184	29054.5	29166.3
		50	45461.5	0.162	0.889	3432.2	17329	45412.1	45537.3
		60	64523.4	0.023	0.187	386.1	3350	64512.0	64541.1
		80	111415.9	0.106	0.531	1959.4	9690	111375.6	111482.7
		100	171735.7	0.076	0.593	1118.8	8578	171716.0	171760.2
		125	255674.4	39.877	398.580	89728.4	895011	255646.4	255740.3
		150	403667.3*	215.434	1800.006	163179.2	879154	403520.7	403704.9
		200	693071.6*	853.774	1800.016	527923.7	1305925	692432.4	693109.5
		300	1538960.3*	900.008	1800.015	521764.8	1169941	1537978.4	1538960.3
		400	2749664.2*	1218.873	1800.016	767075.6	1229435	2748145.6	2749692.9
	0.75	5	746.6	0	0	3.3	6	739.1	746.6
		10	2121.7	0	0	2.5	10	2119.8	2121.8
		15	4146.9	0	0	16.2	48	4126.1	4161.7
		20	7106.8	0	0	27.2	82	7089.1	7121.1
		30	16017.8	0.005	0.016	98.1	388	15992	16037.3
		40	29054.4	0.012	0.062	253.6	1418	29036.8	29091.9
		50	45400.6	0.019	0.141	311.5	2555	45386.6	45419.3
		60	64523.4	0.035	0.314	386.1	3350	64512.0	64541.1
		80	111397.5	0.092	0.296	990.4	3494	111367.2	111436.7
		100	171735.7	0.117	0.889	1118.8	8578	171716.0	171760.2
		125	255646.5	0.027	0.140	227.3	1158	255638.4	255657.7
		150	403527.0	38.232	381.108	75263.8	745503	403511.4	403564.6
		200	692542.7*	315.174	1800.015	216341.9	1305925	692404.5	692580.6
		300	1538291.3*	360.007	1800.010	225155.2	1164078	1537951.7	1538291.3
		400	2748534.3*	720.030	1800.014	405874.7	1161421	2748116.9	2748563.0
0.75	0.25	5	686.2	0	0	0	0	686.2	686.2
		10	2149.6	0	0	8	8	2114.4	2149.6
		15	4085.8	0	0	26.4	88	4041.9	4088.8
		20	7009.2	0	0	48.3	91	6969.3	7014.0
		30	15857.5	0.005	0.016	364.4	1211	15785.4	15870.1
		40	28867.7	0.033	0.109	1591.8	5379	28763.8	28904.5
		50	45132.3	0.229	1.560	6126.8	33896	45025.0	45182.3
		60	64216.8	40.904	384.072	61920.5	422681	64070.2	64280.3
		80	110970.7	93.373	399.891	157080.8	466635	110776.9	111112.3
		100	171022.8	30.548	194.766	82912.4	442563	170946.1	171117.9
		125	254824.0	182.845	1316.089	250666.6	1085779	254731.1	254946.7
		150	402652.9*	637.981	1800.004	409470.9	1057196	402301.5	402757.0
		200	691278.1*	905.081	1800.016	477968.9	1068483	690826.1	691335.1
		300	1535957.0*	721.028	1800.013	422401.7	1149636	1535569.3	1535999.2
		400	2746344.2*	1260.129	1800.015	733333.2	1173758	2744942.3	2746345.9
	0.5	5	686.2	0	0	0	0	686.2	686.2
		10	2108.3	0	0	5.6	8	2085.2	2108.3
		15	4012.6	0	0	5.2	13	4004.5	4012.6
		20	6993.0	0	0	39.2	90	6960.9	6996.4
		30	15830.2	0.009	0.047	268.7	1211	15774.6	15838.5
		40	28806.1	0.028	0.140	803.6	4040	28741.4	28830.7

	50	45064.2	0.097	0.640	2018.6	12909	45003.9	45097.6
	60	64157.1	2.986	25.615	19652.4	142670	64057.9	64212.3
	80	110872.4	48.8796	415.350	81409.3	466635	110753.9	110969.3
	100	171022.8	34.369	216.981	82912.4	442563	170946.1	171117.9
	125	254824.0	194.903	1371.099	250666.6	1085779	254731.1	254946.7
	150	402358.8*	292.988	1800.007	200120.8	1037084	402277.0	402462.9
	200	691252.2*	907.673	1800.012	501208.9	1078783	690826.1	691335.1
	300	1535957.0*	721.856	1800.012	419050.6	1144704	1535569.3	1535999.2
	400	2746344.2*	1260.234	1800.010	708943.9	1113368	2744942.3	2746345.9
0.75	5	686.2	0	0	0	0	686.2	686.2
	10	2071.8	0	0	3.2	8	2058.1	2071.8
	15	4012.6	0	0	5.2	13	4004.5	4012.6
	20	6978.6	0.002	0.015	30.2	62	6952.7	6981.6
	30	15744.2	0	0	13.3	39	15741.5	15745.8
	40	28806.1	0.041	0.203	803.6	4040	28741.4	28830.7
	50	45040.4	0.047	0.203	727.7	2943	44994.9	45068.3
	60	64113.9	0.551	3.994	5385.4	35033	64046.4	64155.8
	80	110743.7	0.117	0.795	1241.1	8753	110720.7	110766.9
	100	171022.8	37.381	234.031	82912.4	442563	170946.1	171117.9
	125	254824.0	201.061	1370.498	250666.6	1085779	254731.1	254946.7
	150	402358.8*	300.817	1800.001	200194.1	1037084	402277.0	402462.9
	200	691118.6*	730.1119	1800.006	407748.7	1037044	690815.8	691201.8
	300	1535737.3*	542.661	1800.006	302076.6	1122950	1535558.6	1535779.5
	400	2745266.7*	540.357	1800.013	271465.8	924076	2744900.1	2745268.4

Table 3
Results of six instances with 200 jobs.

No.	BAB	t_{BAB}	ND	LB	R_{LB}
1	667270	0.140	717	/	/
2	731257	100.956	288414	/	/
3	760512	1800.006	770139	760286	0. 03%
4	670120	1800.006	982530	669159	0. 14%
5	690566	1800.006	998643	689402	0. 17%
6	671753	1800.005	1037044	671183	0. 08%

7.2.1. Comparison of the dynamic programming and the branch-and-bound algorithm

According to the obtained results, we can draw a conclusion that these two algorithms are complementary to each other. To a certain extent, the dynamic programming algorithm outperforms the branch-and-bound algorithm. It can solve problems with up to 1500 jobs when $\alpha=\beta=0.25$. Even when $\alpha=\beta=0.75$, which is the worst case, it can still solve problems with 150 jobs. On the other hand, the branch-and-bound algorithm can solve optimally problems with up to 300 jobs when $\alpha=0.25$and $\beta=0.75$, and it can only solve problems with 125 jobs in the others cases. However, when $n\geq 200$, there are some instances in which an optimal solution cannot be found by the dynamic programming algorithm due to memory problems, while in this case, the branch-and-bound algorithm is more likely to find an optimal solution. Even if it cannot, it can always find a near optimal solution within the given time limit. For example, when $n=200$ and $\alpha=\beta=0.75$, there are 6 out of 10 instances in which optimal solutions cannot be obtained by the dynamic programming, their solutions obtained by the branch-and-bound are given in Table 3. If the branch-and-bound cannot find the optimal solution either, the performance of the solution is indicated by the error to the lower bounds at the root of tree, i.e., $R_{LB}=(BAB-LB)/LB$, where BAB is the objective value, LB is the lower bound and R_{LB} is the error (when an optimal solution can be found, it is not necessary to calculate LB and R_{LB}, so it is denoted by the symbol "/"). From Table 3, we know that these objective values are very close to the lower bounds and are even closer to the optimal values. The results from Tables 2 and 3 also show that the lower bound and the initial solution proposed in Section 6.2 are good since the values are close to the optimal ones.

7.2.2. Impact of the parameters α and β

According to Theorems 2 and 3, there are some cases in which the SPT algorithm is optimal. Table 4 shows the number of instances solved optimally by this algorithm. From the results in Table 4, we can conclude that the number of instances solved optimally by the SPT algorithm increases with the increase of β. This conclusion holds obviously since the time interval in which the maintenance can be scheduled becomes longer with the increase of β. However, the effect of the value of α on the number of instances is not clear.

In the following, we try to analyze how the values of α and β affect the performance of these algorithms. We start with the dynamic programming. From the calculation of the time complexity in Section 5, we can easily deduce that the computation time is an increasing function of α and β for a given instance theoretically, but the results in Table 1 do not seem to show such relationships. The reason could be that the increase of β increases the probability of a given instance being solved optimally by the SPT algorithm. If it happens, the computation time is zero, so the average computation time may decrease on the contrary. This conclusion can be confirmed by examining

Table 4
The number of instances solved optimally by the SPT algorithm.

n	$\alpha=0.25$			$\alpha=0.5$			$\alpha=0.75$		
	$\beta=0.25$	$\beta=0.5$	$\beta=0.75$	$\beta=0.25$	$\beta=0.5$	$\beta=0.75$	$\beta=0.25$	$\beta=0.5$	$\beta=0.75$
5	0	1	6	1	1	2	10	10	10
10	2	6	8	2	7	7	0	3	6
15	4	6	8	1	3	4	3	6	6
20	2	7	9	0	1	2	1	2	3
30	3	4	5	1	3	4	2	3	6
40	4	4	6	3	5	7	1	3	3
50	1	3	6	1	3	6	1	3	4
60	3	7	10	1	6	6	2	2	4
80	2	4	7	2	2	3	2	4	7
100	0	6	7	5	5	5	1	1	1
125	5	6	7	5	5	6	2	2	2
150	0	4	6	2	6	7	2	4	4
200	3	6	8	1	5	8	3	3	4
300	0	4	8	3	4	7	4	4	5
400	2	5	7	2	2	5	1	1	5

Table 5
Results of a set of instances with 60 jobs.

α	β											
	0.25				0.5				0.75			
	DP/BAB	t_{DP}	t_{BAB}	*ND*	*DP/BAB*	t_{DP}	t_{BAB}	*ND*	*DP/BAB*	t_{DP}	t_{BAB}	*ND*
0.25	72986	0.015	0.006	157	72986	0.047	0.012	157	72897	0	0	0
0.5	72336	0.047	0.134	3350	72336	0.093	0.187	3350	72336	0.125	0.314	3350
0.75	71724	0.063	0.010	283	71724	0.124	0.019	283	71724	0.188	0.030	283

Table 6
The maximum size of solved problems by the dynamic programming algorithm.

α	β		
	0.25	0.5	0.75
0.25	1500	700	500
0.5	700	300	200
0.75	400	200	150

the results given in Table 5, which are generated for one of the ten data sets when $n=60$ in Table 2, and the other instances in Table 2 share the same feature. With respect to the memory requirement, the outcome in Table 6 is consistent with the discussion in Section 5, i.e., the required memory increases and consequently the maximum size of problems which can be handled decreases with the increase of α or β.

For the branch-and-bound schema, it is difficult to find a relationship between its performance and the value of α or β from Table 2, since the relationship may be covered up by whether the SPT algorithm is optimal or not. Similar to the dynamic programming algorithm, in Table 5, we give the detailed results for the same instances as those in previous paragraph. From these results, we can conclude that the performance of this algorithm has no clear relationship with the values of α. However, it is related to β. In fact, if the increase of β cannot lead to an optimal solution by the SPT algorithm, the computation time will increase, while the number of nodes are unaffected. The reason why the computation time becomes longer can be explained as follows: the increase of β leads to a longer time period for the maintenance, which further increases the number of possible values of B, therefore, it will take more time to find the lower bound of all the nodes.

8. Conclusions

In the paper, we consider a single machine scheduling problem with a flexible maintenance activity. The objective is to find a feasible job sequence and a starting time of the maintenance to minimize the total completion time. Firstly, basic properties of an optimal solution are given. Based on those properties, it is shown that the proposed SPT algorithm is optimal for the resumable case, while for the nonresumable case, the conditions under which the algorithm is optimal are presented. In addition, it is shown that the relative error bound of SPT algorithm cannot be improved even if the starting time of the maintenance is relaxed. Furthermore, to find the optimal solution, a dynamic programming algorithm and a branch-and-bound algorithm are proposed. Finally, experiments using randomly generated problems are conducted to show the effectiveness and complementarity of these algorithms. In the future, we will extend this model to other complicated problems in practical production.

As reviewed previously, He et al. [10] proposed a polynomial time approximation scheme for the problem with a deterministic maintenance. Thus, we doubt whether there is also a polynomial time approximation scheme when the maintenance is flexible, which is an open problem and a potential direction of our future research.

Acknowledgements

The authors thank the partial support of the Natural Science Foundation of China under Grant numbers 70631003 and 60736026, UK Engineering and Physical Science Research Council under Grant number EP/F024606/1, and the Doctoral Fund of Ministry of Education of China under Grant number

200803590007. And the authors are also thankful to anonymous referees for their constructive comments which have led to a substantial improvement to the paper.

Appendix A. Proof of Theorem 4

Let σ be a schedule generated by the SPT algorithm, δ be the idle time, L and R be the sets of jobs scheduled before and after the maintenance, respectively. Accordingly, let σ^* be an optimal schedule, X and Y be the sets of the first $|L|$ jobs and the remaining jobs in σ^*, respectively (clearly$|L|=|X|$ and $|R|=|Y|$). Since set L contains the smallest $|L|$ jobs, then we have $\sum_{j\in X}p_j \geq \sum_{j\in L}p_j$, so the first job in set Y is scheduled after the maintenance in schedule σ^*, as shown in Fig. A.1. Furthermore, if SPT algorithm is not optimal for this case, then set L and set X are different, therefore there must exist jobs which are scheduled in set L of schedule σ and set Y of schedule σ^*. Note that the jobs in set Y are in SPT order, so these jobs are scheduled firstly in set Y, therefore we get $\sum_{J_j\in L}p_j+p_{|L|+1} \leq \sum_{J_j\in X}p_j+p_{[|L|+1]}(\sigma*)$, further we have $C_{|L|+1}(\sigma)=\sum_{J_j\in L}p_j +p_{|L|+1}+\delta+r\leq \sum_{J_j\in X}p_j \quad +p_{[|L|+1]}(\sigma*)+r+\delta \quad \leq C_{[|L|+1]}(\sigma*)+\delta$. Similarly, we can derive that $C_j(\sigma)\leq C_{[j]}(\sigma^*)+\delta$, for $\forall j=|L|+1,\ldots,n$, then we have $f_P(R,\sigma)\leq f_P(Y,\sigma^*)+|R|\delta$, where $f_P(M,\pi)$ is the total completion time of the jobs which belongs to set M in scheduleπ. It is obvious that $C_{|L|}(\sigma)+\delta=s\leq C_{[|L|]}(\sigma^*)$, i.e., $C_{|L|}(\sigma)\leq C_{[|L|]}(\sigma^*)-\delta$, further we have $f_P(L,\sigma)\leq f_P(X,\sigma^*)-\delta$. Therefore, the following relation holds:

$$f_P(\sigma)=f_P(L,\sigma)+f_P(R,\sigma)\leq f_P(X,\sigma^*)+f_P(Y,\sigma^*) +(|R|-1)\delta=f_P(\sigma^*)+(|R|-1)\delta \tag{A.1}$$

From the above reasoning process, we know that in schedule σ, $p_{|L|+1}>\delta$, then we get $p_j>\delta$ for all $|L|+1<j\leq n$ since jobs in set R are in SPT order. Therefore we have $f_P(R,\sigma^*)\geq \delta+2\delta+\ldots|R|\delta =|R|(|R|+1)\delta/2$. Suppose J_i is the job which is scheduled in set L of schedule σ and in set Y of schedule σ^*, then we have $f_P(L,\sigma^*)\geq C_i(\sigma^*)>s>\delta$. Therefore the following relation holds:

$$f_P(\sigma^*)=f_P(L,\sigma^*)+f_P(R,\sigma^*)\geq \frac{(|R|(|R|+1)+2)\delta}{2} \tag{A.2}$$

Combining (A.1) and (A.2), we can obtain the following relative error bound:

$$R_{SPT}=\frac{f_P(\sigma)-f_P(\sigma^*)}{f_P(\sigma^*)}\leq \frac{2(|R|-1)}{|R|(|R|+1)+2} \tag{A.3}$$

Taking the derivative of the function $y=2(x-1)/(x(x+1)+2)$ within the domain of $1\leq x\leq n-1$, we obtain

$$\frac{dy}{dx}=-\frac{2(x-3)(x+1)}{(x(x+1)+2)^2}\begin{cases}>0, & x<3\\ =0 & x=3\\ <0, & x>3\end{cases}$$

From the derivatives, we can see that y increases with x when $x<3$, decreases when $x>3$, and reaches its maximum 2/7 when $x=3$, i.e., when $|R|=3$, $R_{SPT}=2/7$. Also, it is easy for us to obtain $R_{SPT}=0$ when $|R|=1$. This is consistent with Theorem 3.

Appendix B. Proof of Proposition 8

By comparing γ_2 to θ_2, we have $\gamma_2=\theta_2$. In the following, we try to establish the relationship between γ_1 and θ_1.

If $x=|L|$, then $\gamma_1-\theta_1=B-A_{|L|}\geq 0$. And if $x\leq |L|-1$, then

$$\begin{aligned}\gamma_1-\theta_1 &= B-A_{|L|}+(|L|-x)(r+p_{|L|+1})+\frac{1}{p_x}(p_{|L|+1}+r)(A_{x-1}+p_{|L|+1}-B)\\ &\quad -\frac{1}{p_{|L|}}(r+p_{|L|+1})(A_{|L|-1}+p_{|L|+1}-B)\geq (|L|-x)(r+p_{|L|+1})\\ &\quad +\frac{1}{p_x}(p_{|L|+1}+r)(A_{x-1}+p_{|L|+1}-B)-\frac{1}{p_{|L|}}(r+p_{|L|+1})(A_{|L|-1}+p_{|L|+1}-B)\\ &=(r+p_{|L|+1})\left(|L|-x+\frac{1}{p_x}(A_{x-1}+p_{|L|+1}-B)-\frac{1}{p_{|L|}}(A_{|L|-1}+p_{|L|+1}-B)\right)\\ &=(r+p_{|L|+1})\left((A_x+p_{|L|+1}-B)\left(\frac{1}{p_x}-\frac{1}{p_{|L|}}\right)+\frac{1}{p_{|L|}}(A_x+(|L|-x-1)p_{|L|}-A_{|L|-1})\right)\end{aligned}$$

Since $p_{x+1}\leq p_{x+2}\leq\ldots\leq p_{|L|-1}\leq p_{|L|}$, we have $A_x+(|L|-x-1)\times p_{|L|}-A_{|L|-1}\geq 0$. Note that $A_x+p_{|L|+1}-B>0$ and $(1/p_x)-(1/p_{|L|})\geq 0$, therefore we have $\gamma_1-\theta_1>0$. Furthermore we have $LB_5=\min\{\gamma_1,\gamma_2\}\geq LB_3$.

Appendix C. Proof of Proposition 9

We proof this proposition by comparing γ_1 and γ_2 to LB_4. By definitions in Proposition 5 and 7, we have

$$\begin{aligned}\gamma_2-LB_4 &= \frac{1}{p_{|L|+2}}(r+p_{|L|+1}-p_{|L|+2})(A_{|L|}-B)-\frac{(A_{|L|}-B)r}{p_{|L|+1}}\\ &=\left(\frac{1}{p_{|L|+2}}-\frac{1}{p_{|L|+1}}\right)(A_{|L|}-B)(r+p_{|L|+1})\geq 0\end{aligned}$$

In the following, we try to establish the relationship between γ_1 and LB_4. If $x=|L|$, then

$$\begin{aligned}\gamma_1-LB_4 &= B-A_{|L|}+\frac{1}{p_{|L|}}(p_{|L|+1}+r)(A_{|L|-1}+p_{|L|+1}-B)-\frac{(A_{|L|}-B)r}{p_{|L|+1}}\\ &=(A_{|L|}+p_{|L|+1}-B)(r+p_{|L|+1})\left(\frac{1}{p_{|L|}}-\frac{1}{p_{|L|+1}}\right)\geq 0\end{aligned}$$

Otherwise if $x\leq |L|-1$, then

$$\begin{aligned}\gamma_1-LB_4 &= B-A_{|L|}+(|L|-x)(r+p_{|L|+1})+\frac{1}{p_x}(p_{|L|+1}+r)(A_{x-1}+p_{|L|+1}-B)\\ &\quad -\frac{(A_{|L|}-B)r}{p_{|L|+1}}\geq r+p_{|L|+1}+\frac{1}{p_x}(r+p_{|L|+1})(A_{x-1}+p_{|L|+1}-B)\\ &=\frac{1}{p_x}(r+p_{|L|+1})(A_x+p_{|L|+1}-B)>0\end{aligned}$$

Therefore we have $LB_5=\min\{\gamma_1,\gamma_2\}\geq LB_4$.

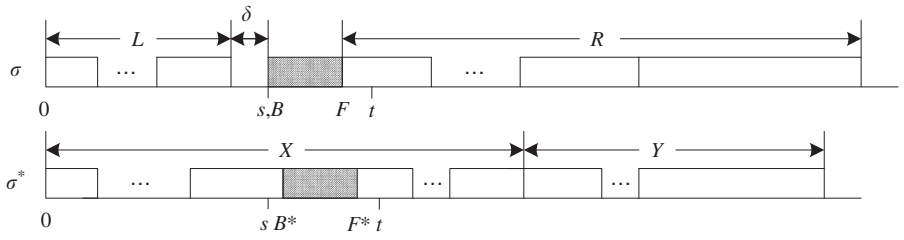

Fig. A.1. Illustration of two schedules.

Appendix D. Proof of $LB_8 < LB_6$

Since $p_i > \delta$ for $\forall |L|+1 \le i \le n$, we have

$$\sum_{i=1}^{n} A_i \ge \sum_{i=|L|+1}^{n} A_i = \frac{|R|(|R|+1)}{2}\delta, \tag{D.1}$$

and further we have

$$f_\varphi(SPT) = \sum_{i=1}^{n} A_i + |R|(r+\delta) > \frac{|R|(|R|+3)}{2}\delta. \tag{D.2}$$

From Eq. (D.2), we can further obtain

$$LB_8 = \frac{|R|(|R|+1)+2}{|R|(|R|+3)} f_\varphi(SPT) = f_\varphi(SPT) - \frac{2(|R|-1)}{|R|(|R|+3)} f_\varphi(SPT)$$
$$< f_\varphi(SPT) - \frac{2(|R|-1)}{|R|(|R|+3)} \times \frac{|R|(|R|+3)}{2}\delta = f_\varphi(SPT) - (|R|-1)\delta = LB_6$$

References

[1] Lee C-Y, Lei L, Pinedo M. Current trends in deterministic scheduling. Annals of Operations Research 1997;70:1–41.
[2] Sanlaville E, Schmidt G. Machine scheduling with availability constraints. Acta Informatica 1998;35:795–811.
[3] Schmidt G. Scheduling with limited machine availability. European Journal of Operational Research 2000;121:1–15.
[4] Ma Y, Chu C, Zuo C. A survey of scheduling with deterministic machine availability constraints. Computers & Industrial Engineering 2010;58:199–211.
[5] Adiri I, Bruno J, Frostig E, Rinnooy Kan AHG. Single machine flow-time scheduling with a single breakdown. Acta Informatica 1989;26:679–96.
[6] Lee C-Y, Liman SD. Single machine flow-time scheduling with scheduled maintenance. Acta Informatica 1992;29:375–82.
[7] Lee C-Y. Machine scheduling with an availability constraint. Journal of Global Optimization 1996;9:395–416.
[8] Sadfi C, Penz B, Rapine C, Błażewicz J, Formanowicz P. An improved approximation algorithm for the single machine total completion time scheduling problem with availability constraints. European Journal of Operational Research 2005;161:3–10.
[9] Breit J. Improved approximation for non-preemptive single machine flow-time scheduling with an availability constraint. European Journal of Operational Research 2007;183:516–24.
[10] He Y, Zhong W, Gu H. Improved algorithms for two single machine scheduling problems. Theoretical Computer Science 2006;363:257–65.
[11] Wang G, Sun H, Chu C. Preemptive scheduling with availability constraints to minimize total weighted completion times. Annals of Operations Research 2005;133:183–92.
[12] Kacem I, Chu C. Worst-case analysis of the WSPT and MWSPT rules for single machine scheduling with one planned setup period. European Journal of Operational Research 2008;187:1080–9.
[13] Kacem I. Approximation algorithm for the weighted flow-time minimization on a single machine with a fixed non-availability interval. Computers & Industrial Engineering 2008;54:401–10.
[14] Kacem I, Chu C, Souissi A. Single-machine scheduling with an availability constraint to minimize the weighted sum of the completion times. Computers & Operations Research 2008;35:827–44.
[15] Kacem I, Chu C. Efficient branch-and-bound algorithm for minimizing the weighted sum of completion times on a single machine with one availability constraint. International Journal of Production Economics 2008;112:138–50.
[16] Mosheiov G, Sarig A. Scheduling a maintenance activity to minimize total weighted completion-time. Computers and Mathematics with Applications 2009;57:619–23.
[17] Mosheiov G, Sarig A. A note: Simple heuristics for scheduling a maintenance activity on unrelated machines. Computers & Operations Research 2009;36:2759–62.
[18] Lee C-Y, Chen ZL. Scheduling jobs and maintenance activities on parallel machines. Naval Research Logistics 2000;47:145–65.
[19] Levin A, Mosheiov G, Sarig A. Scheduling a maintenance activity on parallel identical machines. Naval Research Logistics 2009;56:33–41.
[20] Allaoui H, Lamouri S, Artiba A, Aghezzaf E. Simultaneously scheduling *n* jobs and the preventive maintenance on the two-machine flow shop to minimize the makespan. International Journal of Production Economics 2008;112:161–7.
[21] Qi X, Chen T, Tu F. Scheduling the maintenance on a single machine. Journal of the Operational Research Society 1999;50:1071–8.
[22] Qi X. A note on worst-case performance of heuristics for maintenance scheduling problems. Discrete Applied Mathematics 2007;155:416–22.
[23] Sbihi M, Varnier C. Single-machine scheduling with periodic and flexible periodic maintenance to minimize maximum tardiness. Computers & Industrial Engineering 2008;55:830–40.
[24] Graves GH, Lee C-Y. Scheduling maintenance and semiresumable jobs on a single machine. Naval Research Logistics 1999;46:845–63.
[25] Akturk MS, Ghosh JB, Gunes ED. Scheduling with tool changes to minimize total completion time: a study of heuristics and their performance. Naval Research Logistics 2003;50:15–30.
[26] Akturk MS, Ghosh JB, Gunes ED. Scheduling with tool changes to minimize total completion time: basic results and SPT performance. European Journal of Operational Research 2004;157:784–90.
[27] Yang DL, Hung CL, Hsu CJ, Chern MS. Minimizing the makespan in a single machine scheduling problem with a flexible maintenance. Journal of the Chinese Institute of Industrial Engineers 2002;19:63–6.
[28] Chen JS. Single-machine scheduling with flexible and periodic maintenance. Journal of the Operational Research Society 2006;57:703–10.
[29] Lau HC, Zhang C. Job scheduling with unfixed availability constraints. In: Proceedings of 35th Meeting of the Decision Sciences Institute. Boston, USA; 2004. p. 4401–6.
[30] Chen JS. Optimization models for the machine scheduling problem with a single flexible maintenance activity. Engineering Optimization 2006;38:53–71.
[31] Chen JS. Scheduling of nonresumable jobs and flexible maintenance activities on a single machine to minimize makespan. European Journal of Operational Research 2008;190:90–102.
[32] Xu D, Yin Y, Li H. A note on "scheduling of nonresumable jobs and flexible maintenance activities on a single machine to minimize makespan". European Journal of Operational Research 2009;197:825–7.
[33] Xu D, Sun K, Li H. Parallel machine scheduling with almost periodic maintenance and non-preemptive jobs to minimize makespan. Computers & Operations Research 2008;35:1344–9.
[34] Gao J, Gen M, Sun L. Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm. Journal of Intelligent Manufacturing 2006;17:493–507.
[35] Dell' Amico M, Martello S. Bounds for the cardinality constrained *P*||Cmax problem. Journal of Scheduling 2001;4:123–38.
[36] Liao C-J, Chen C-M, Lin C-H. Minimizing makespan for two parallel machines with job limit on each availability interval. Journal of the Operational Research Society 2007;58:938–47.
[37] Yang D-L, Hsub C-J, Kuo W-H. A two-machine flowshop scheduling problem with a separated maintenance constraint. Computers & Operations Research 2008;35:876–83.
[38] Low C, Ji M, Hsu C-J, Su C-T. Minimizing the makespan in a single machine scheduling problems with flexible and periodic maintenance. Applied Mathematical Modelling 2010;34:334–42.
[39] Xu D, Yin Y, Li H. Scheduling jobs under increasing linear machine maintenance time. Journal of Scheduling 2010;13:443–9.
[40] Cassady CR, Kutanoglu E. Minimizing job tardiness using integrated preventive maintenance planning and production scheduling. IIE Transactions 2003;35:503–13.
[41] Cassady CR, Kutanoglu E. Integrating preventive maintenance planning and production scheduling for a single machine. IEEE Transactions on Reliability 2005;54:304–9.
[42] Sortrakul N, Nachtmann HL, Cassady CR. Genetic algorithms for integrated preventive maintenance planning and production scheduling for a single machine. Computers in Industry 2005;56:161–8.
[43] Ruiz R, García-Díaz JC, Maroto C. Considering scheduling and preventive maintenance in the flowshop sequencing problem. Computers & Operations Research 2007;34:3314–30.
[44] Berrichi A, Amodeo L, Yalaoui F, Châtelet E, Mezghiche M. Bi-objective optimization algorithms for joint production and maintenance scheduling: application to the parallel machine problem. Journal of Intelligent Manufacturing 2009;20:389–400.
[45] Kaabi J, Varnier C, Zerhouni N. Heuristics for scheduling maintenance and production on a single machine. In: Proceedings of IEEE Conference on Systems, Man and Cybernetics. Hammamet, Tunisia; 2002.
[46] Youssef H, Brigitte C-M, Noureddine Z. Lower bounds and multiobjective evolutionary optimization for combined maintenance and production scheduling in job shop. In: Proceedings of Emerging Technologies and Factory Automation. Lisbon, Portugal; 2003. p. 95–100.
[47] Chen W-J, Liao C-J. Scheduling with different maintenance policies in a textile company. Journal of Quality in Maintenance Engineering 2005;11:43–52.
[48] Posner ME. Minimizing weighted completion times with deadlines. Operations Research 1985;33:562–74.
[49] Belouadah H, Posner ME, Potts CN. Scheduling with release dates on a single machine to minimize total weighted completion time. Discrete Applied Mathematics 1992;36:213–31.
[50] Webster S. Weighted flow time bounds for scheduling identical processors. European Journal of Operational Research 1995;80:103–11.