J Sched (2010) 13: 443–449
DOI 10.1007/s10951-010-0182-0

Scheduling jobs under increasing linear machine maintenance time

Dehua Xu · Yunqiang Yin · Hongxing Li

Published online: 4 June 2010

Abstract Although scheduling problems with machine availability have attracted many researchers' attention, most of the past studies are mainly focused on one or several prefixed machine maintenance activities. In this research, we assume that the time needed to perform one maintenance activity is an increasing linear function of the total processing time of the jobs that are processed after the machine's last maintenance activity. We consider two scheduling problems with such maintenance requirement in this paper. The first problem is a parallel machine scheduling problem where the length of the time interval between any two consecutive maintenance activities is between two given positive numbers. The objective is to minimize the maintenance makespan, i.e., the completion time of the last finished maintenance. The second problem is a single machine scheduling problem where the length of the time interval between any two consecutive maintenance activities is fixed and the objective is to minimize the makespan, i.e., the completion time of the last finished job. We propose two approximation algorithms for the considered problems and analyze their performances.

Keywords Scheduling · Maintenance · Linear function · Approximation algorithm

D. Xu · Y. Yin · H. Li (✉)
School of Mathematical Sciences, Beijing Normal University, Beijing 100875, China
e-mail: lhxqx@bnu.edu.cn

Y. Yin
e-mail: yinyunqiang@mail.bnu.edu.cn

D. Xu
e-mail: xudehua@mail.bnu.edu.cn

1 Introduction

Although scheduling problems with machine availability have attracted many researchers' attention, most of the past studies are mainly focused on one or several prefixed machine maintenance activities (see, e.g., Sanlaville and Schmidt 1998; Schmidt 2000 and Lee 2004). However, in the real world, the maintenance time of a machine may vary depending on the amount of jobs that are processed after its last maintenance. Intuitively, the machine that has more jobs processed should have longer maintenance time, since it may be in a worse condition after a long processing time and needs more time to be maintained. For example, in transportation companies, the truck which has a long journey usually takes more time to be maintained than that which has a short journey.

As linear and piecewise linear functions are relatively easy to manipulate and are usually adopted to approximate nonlinear functions, we suppose that the amount of time needed to perform one maintenance activity on a machine is an increasing linear function of the total processing time of the jobs that are processed after its last maintenance. We consider two scheduling problems with such maintenance requirements in this paper, one is a parallel machine scheduling problem and the other is a single machine scheduling problem. To the best of our knowledge, such problems have not been studied in the literature.

We use the worst-case bound to measure the performance quality of an approximation algorithm. Specifically, for an instance I of a minimization problem, let $C^A_{\max}(I)$ denote the value produced by an approximation algorithm A, and $C^*_{\max}(I)$ the minimum value. Then the *worst-case bound* R_A of algorithm A is defined as the smallest number ρ such that for any instance I, $C^A_{\max}(I) \leq \rho C^*_{\max}(I)$. If a polynomial time approximation algorithm A can achieve worst-case bound ρ, we say that A is a polynomial time ρ-approximation algorithm.

The rest of the paper is organized as follows. In Sect. 2, we give formal formulations of the problems under consideration. In Sect. 3, we propose one approximation algorithm for each of the scheduling problems. In Sect. 4, the performances of the algorithms are analyzed. Finally, in Sect. 5 we give some concluding remarks.

2 Problem formulation

The parallel machine scheduling problem considered in this paper can be formally described as follows. There are n independent jobs $J_1, J_2, \ldots, J_n$ to be processed on m parallel identical machines $P_1, P_2, \ldots, P_m$. The processing time of job J_i is p_i. All jobs are nonpreemptive and are available at time zero. We assume that each machine can process at most one job at a time and that each job can be processed on at most one machine at a time. All machines have the same maintenance requirement: the length of the time interval between any two consecutive maintenance activities is within a prefixed interval $[T, T']$, where T and T' are two positive real numbers such that $T' - T \geq 0$ and $T' \geq p_i$ for $i = 1, 2, \ldots, n$. The amount of time needed to perform one maintenance activity on a machine is an increasing linear function $T_M(t) = a + bt$ of the total processing time t of the jobs that are processed after its last maintenance, where a and b are nonnegative real numbers. We assume that all machines have just finished their maintenances at time zero and must be maintained after their processing. The objective is to minimize the *maintenance makespan* $MC_{\max}$, i.e., the completion time of the last finished maintenance. Extending the well-known three field $\alpha|\beta|\gamma$ classification scheme suggested by Graham et al. (1979), we describe this problem as $Pm, MS[T, T'], T_M(t) = a + bt||MC_{\max}$.

Figure 1 presents a schedule on machine P_i for $Pm, MS[T, T'], T_M(t) = a + bt||MC_{\max}$, where $J_l^{(ij)}$ denotes the lth job assigned to the jth working interval B_{ij} of machine P_i, $T^{(ij)}$ denotes the length of the time of B_{ij}, $M^{(ij)}$ denotes the jth maintenance activity of machine P_i, and $T_M^{(ij)}$ denotes the length of the time of $M^{(ij)}$. $T^{(ij)} \in [T, T']$ and $T_M^{(ij)} = a + b\sum_{l=1}^{i_j} p_l^{(ij)}$ must hold if the above schedule is a feasible schedule.

Using similar terminology, the single machine scheduling problem considered in this paper can be described as $1, MS[T, T], T_M(t) = a + bt, b \leq 1||C_{\max}$, where $C_{\max}$ is the completion time of the last finished job and $T \geq p_i$ for $i = 1, 2, \ldots, n$.

Figure 2 presents a schedule for $1, MS[T, T], T_M(t) = a + bt, b \leq 1||C_{\max}$, where $J_l^{(j)}$ denotes the lth job assigned to the jth working interval B_j of the machine, $M^{(j)}$ denotes the jth maintenance period of the machine, and $T_M^{(j)}$ denotes the length of the time of $M^{(j)}$. $T_M^{(j)} = a + b\sum_{l=1}^{i_j} p_l^{(j)}$ must hold if the above schedule is a feasible schedule.

Recently, Ji et al. (2007) considered the *NP*-hard scheduling problem $1, MS[T, T], T_M(t) \equiv a||C_{\max}$. They proved that the worst-case bound of the classical LPT (Longest Processing Time first) algorithm is 2 and showed that there is no polynomial time approximation algorithm with a worst-case bound less than 2, unless $P = NP$. Xu et al. (2008) considered the *NP*-hard scheduling problem $Pm, MS[T, T'], T_M(t) \equiv a||MC_{\max}$. They proposed a $(2T'/T)$-approximation algorithm, named BFD-LPT, for the problem and showed that there is no polynomial time approximation algorithm with a worst-case bound less than 2, unless $P = NP$. Obviously, our problems are more complex and general, and thus *NP*-hard. However, to the best of our knowledge, there is no approximation algorithm provided and analyzed in the literature.

3 Approximation algorithms

In this section, we introduce the Modified BFD-LPT algorithm and the FFD-LS algorithm for our problems. Before we give these two algorithms, we first present some related algorithms and problems. BFD (Best Fit Decreasing) algorithm and FFD (First Fit Decreasing) algorithm are two efficient approximation algorithms for the one dimensional bin-packing problem, while LPT algorithm and

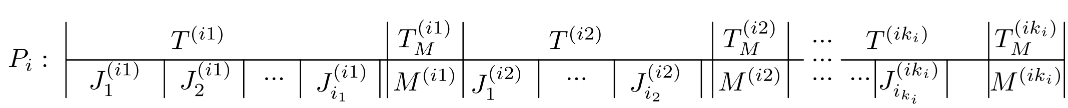

Fig. 1 A schedule on machine P_i for $Pm, MS[T, T'], T_M(t) = a + bt||MC_{\max}$

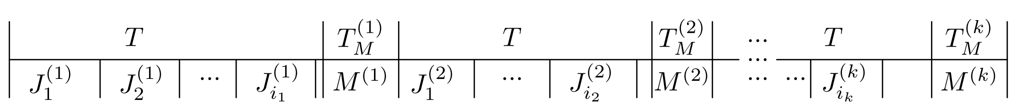

Fig. 2 A schedule for $1, MS[T, T], T_M(t) = a + bt, b \leq 1||C_{\max}$

LS (List Scheduling) algorithm are two classical heuristics for machine scheduling problems. The one dimensional bin-packing problem and the above four algorithms can be formally described as follows.

One dimensional bin-packing problem (see, e.g., Coffman et al. 1997) Given n items $a_1, a_2, \ldots, a_n$, each with a size $s(a_i) \in (0, W]$, we are asked to pack them into a minimum number of W-capacity bins (i.e., partition them into a minimum number m of subsets $B_1, B_2, \ldots, B_m$ such that $\sum_{a_i \in B_j} s(a_i) \leq W, 1 \leq j \leq m$).

Algorithm BFD (see, e.g., Coffman et al. 1997) Sort all the items such that $s(a_1) \geq s(a_2) \geq \cdots \geq s(a_n)$; for $i = 1, 2, \ldots, n$, item a_i is packed in the partially-filled bin B_j with the highest level $level(B_j)$, among bins with $level(B_j) \leq W - s(a_i)$, where $level(B_j)$ is the sum of the size of the items in bin B_j, ties are broken in favor of lower index. If no such bin exists, we start a new bin with a_i as its first item.

Algorithm FFD (see, e.g., Coffman et al. 1997) Sort all the items such that $s(a_1) \geq s(a_2) \geq \cdots \geq s(a_n)$; for $i = 1, 2, \ldots, n$, item a_i is packed in the first (lowest indexed) bin into which it will fit, i.e., if there is any partially-filled bin B_j with $level(B_j) \leq W - s(a_i)$, we place a_i in the lowest-indexed bin having this property. Otherwise, we start a new bin with a_i as its first item.

Algorithm LPT (see, e.g., Ji et al. 2007) Sort all the jobs such that $p_1 \geq p_2 \geq \cdots \geq p_n$; then process the jobs consecutively as early as possible.

Algorithm LS (see, e.g., Graham 1966) Put all the jobs on a list in arbitrary order; then process the jobs consecutively as early as possible.

Since the one dimensional bin-packing problem is one of the oldest and most thoroughly studied problems in the field of combinatorial optimization, many researches are focused on applications of bin-packing algorithms and related results to scheduling problems (see, e.g., Coffman et al. 1978; Ji et al. 2007). It is interesting to see that if $m = 1$, $T = T'$, $a > 0$, and $b = 0$, our problem $Pm, MS[T, T'], TM(t) = a + bt||MC_{\max}$ is essentially the same as the one dimensional bin-packing problem and that if $T = T'$, $a > 0$, and $b = 0$, the scheduling problem $Pm, MS[T, T'], TM(t) = a + bt||MC_{\max}$ can be viewed as the following *one dimensional bin-packing problem with m packing lines*.

One dimensional bin-packing problem with m packing lines There are n items to be packed on m packing lines, where there are infinite many T-capacity bins available. Let the number of used bins in packing line i be L_i. The objective is to pack all the items into bins such that $\max\{L_i \mid 1 \leq i \leq m\}$ is minimum.

The underlying ideas of our algorithms are straightforward. Taking the Modified BFD-LPT algorithm as an example, we think of each interval between two consecutive maintenance activities as a bin and the jobs as items. We first obtain a number of used bins by the BFD algorithm. Then we attach a special item, i.e., a maintenance activity, to each used bin. Let each used bin plus the corresponding attached item be viewed as a single job, assign these jobs to the machines by the LPT algorithm.

Given an instance I of $Pm, MS[T, T'], T_M(t) = a + bt||MC_{\max}$: $J_1, J_2, \ldots, J_n$, the processing time of job J_i is p_i. We construct the corresponding instance II of the one dimensional bin-packing problem as follows: There are n items $a_1, a_2, \ldots, a_n$, the size of item a_i is p_i, and the capacity of each bin is T'. Formally, the Modified BFD-LPT algorithm for $Pm, MS[T, T'], T_M(t) = a + bt||MC_{\max}$ can be described as follows.

Algorithm Modified BFD-LPT

Step 1. If $\sum_{i=1}^{n} p_i \geq mT'/2$, go to Step 2; else, schedule the jobs to the machines by the LPT algorithm, perform one maintenance activity according to the maintenance requirement on each machine as early as possible. Stop.

Step 2. Construct the corresponding instance II of the one dimensional bin-packing problem from the scheduling instance I as stated above. Using the BFD algorithm, we obtain k used bins $B_1, B_2, \ldots, B_k$. Let bin B_i be denoted as $(a_1^{(i)}, a_2^{(i)}, \ldots, a_{k_i}^{(i)})$, where k_i is the number of items in B_i and $a_j^{(i)}$ is the jth item assigned to B_i.

Step 3. For $i = 1, 2, \ldots, k$, if $level(B_i) \geq T$, then let $\mathcal{J}_i = (J_1^{(i)}, J_2^{(i)}, \ldots, J_{k_i}^{(i)}, M^{(i)})$; otherwise, let $\mathcal{J}_i = (J_1^{(i)}, J_2^{(i)}, \ldots, J_{k_i}^{(i)}, \emptyset_i, M^{(i)})$, where $J_j^{(i)}$ is the job corresponding to item $a_j^{(i)}$, $\emptyset_i$ is a dummy job with the processing time of $T - \sum_{j=1}^{k_i} p_j^{(i)}$ (when processing a dummy job, the machine waits the corresponding time), and $M^{(i)}$ denotes a maintenance activity with the length of $a + level(B_i)b$.

Step 4. Let $\mathcal{J}_i$ be viewed as a single job with the processing time of $\max\{level(B_i), T\} + a + level(B_i)b$. Assign $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_k$ to the m parallel machines by the LPT algorithm.

The computational time complexities of the BFD algorithm and the LPT algorithm are $O(n^2)$ and $O(n \log n)$, respectively. Step 3 needs $O(n)$ time. So the Modified BFD-LPT algorithm has a computational time complexity $O(n^2)$.

Algorithm FFD-LS

Step 1. Let $T' = T$. Construct the corresponding instance II of the one dimensional bin-packing problem from the scheduling instance I as stated above. Using the FFD algorithm, we obtain k used bins $B_1, B_2, \ldots, B_k$. Let bin B_i be denoted as $(a_1^{(i)}, a_2^{(i)}, \ldots, a_{k_i}^{(i)})$, where k_i is the number of items in B_i and $a_j^{(i)}$ is the jth item assigned to B_i.

Step 2. For $i = 1, 2, \ldots, k$, let $\mathcal{J}_i = (J_1^{(i)}, J_2^{(i)}, \ldots, J_{k_i}^{(i)}, \emptyset_i, M^{(i)})$, where $J_j^{(i)}$ is the job corresponding to item $a_j^{(i)}$, $\emptyset_i$ is a dummy job with the processing time of $T - \sum_{j=1}^{k_i} p_j^{(i)}$, and $M^{(i)}$ denotes a maintenance activity with the length of $a + level(B_i)b$.

Step 3. Let $\mathcal{J}_i$ be viewed as a single job with the processing time of $T + a + level(B_i)b$. Assign $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_{k-1}$ to the machine by the LS algorithm. Assign $\mathcal{J}_k$ to the machine.

The computational time complexity of FFD-LS algorithm is $O(n^2)$ since the complexities of the FFD algorithm and the LS algorithm are $O(n^2)$ and $O(n)$, respectively.

4 Performance analysis

4.1 Performance analysis of the Modified BFD-LPT algorithm for the parallel machine scheduling problem

Before analyzing the Modified BFD-LPT algorithm, we first present some lemmas.

Lemma 1 (Graham 1966) *If the LPT algorithm is used to solve the scheduling problem $Pm||C_{\max}$, then $R_{\text{LPT}} = 4/3 - 1/(3m)$.*

Lemma 2 (Simchi-Levi 1994) *If algorithm $A \in \{FFD, BFD\}$ is used to solve the one dimensional bin-packing problem, then $R_A = 3/2$.*

Now we give an estimate of the worst-case bound of the Modified BFD-LPT algorithm.

Theorem 3 *The worst-case bound of the Modified BFD-LPT algorithm for the scheduling problem $Pm, MS[T, T'], T_M(t) = a + bt||MC_{\max}$ is at most $\max\{2(T' + a)/(T + a), 4\}$.*

Proof Let $MC_{\max}^{\text{M-BFD-LPT}}$ and $MC_{\max}^*$ denote the maintenance makespan derived by the Modified BFD-LPT algorithm and the optimal maintenance makespan, respectively. Assume that we obtain k used bins according to Step 2 of the Modified BFD-LPT algorithm, while the optimal (minimum) number of bins is k^*.

If $\sum_{i=1}^n p_i < mT'/2$, it is easy to see that the makespan derived by the LPT algorithm is at most T'. Note that by Lemma 1, the LPT algorithm has a worst-case performance bound of $\frac{4}{3} - \frac{1}{3m}$ for $Pm||C_{\max}$, i.e., $C_{\max}^{\text{LPT}}/C_{\max}^* \le \frac{4}{3} - \frac{1}{3m}$, where $C_{\max}^{\text{LPT}}$ and $C_{\max}^*$ denote the LPT makespan and optimal makespan for $Pm||C_{\max}$, respectively. So we have

$$\begin{aligned}\frac{MC_{\max}^{\text{M-BFD-LPT}}}{MC_{\max}^*} &\le \frac{T' + a + bC_{\max}^{\text{LPT}}}{T + a + bC_{\max}^*} \le \max\left\{\frac{T'+a}{T+a}, \frac{C_{\max}^{\text{LPT}}}{C_{\max}^*}\right\} \\ &\le \max\left\{\frac{T'+a}{T+a}, \frac{4}{3} - \frac{1}{3m}\right\}.\end{aligned}$$

If $\sum_{i=1}^n p_i \ge mT'/2$, we consider the following three cases.

Case 1: $0 < k \le m$. Note that $MC_{\max}^{\text{LPT}} \le T' + a + bT'$ and $MC_{\max}^* \ge T + a + b\sum_{i=1}^n p_i/m \ge T + a + bT'/2$, so we have

$$\frac{MC_{\max}^{\text{M-BFD-LPT}}}{MC_{\max}^*} \le \frac{T' + a + bT'}{T + a + bT'/2} \le \max\left\{\frac{T'+a}{T+a}, 2\right\}.$$

Case 2: $m < k \le 2m$. It is easy to see that there are at most two maintenance activities on each machine. So we have $MC_{\max}^{\text{LPT}} \le 2(T' + a + bT')$. Note that $\sum_{i=1}^n p_i > kT'/2$ and $k > m \ge 2$, so we have $MC_{\max}^* \ge T + a + b((kT'/2)/m) > T + a + bT'/2$. Thus,

$$\frac{MC_{\max}^{\text{M-BFD-LPT}}}{MC_{\max}^*} \le \frac{2(T' + a + bT')}{T + a + bT'/2} \le \max\left\{\frac{2(T'+a)}{T+a}, 4\right\}.$$

Case 3: If $k > 2m$. It is easy to see that the maintenance makespan obtained by the Modified BFD-LPT is no more than $\lceil\frac{k}{m}\rceil(T' + a + bT')$ while the optimal maintenance makespan is at least $\lceil\frac{k^*}{m}\rceil(T + a) + b\sum_{i=1}^n p_i/m$. So we have

$$\begin{aligned}\frac{MC_{\max}^{\text{M-BFD-LPT}}}{MC_{\max}^*} &\le \frac{\lceil\frac{k}{m}\rceil(T' + a + bT')}{\lceil\frac{k^*}{m}\rceil(T + a) + b\sum_{i=1}^n p_i/m} \\ &= \frac{\lceil\frac{k}{m}\rceil(T' + a) + \lceil\frac{k}{m}\rceil bT'}{\lceil\frac{k^*}{m}\rceil(T + a) + b\sum_{i=1}^n p_i/m}.\end{aligned}$$

Note that $k/k^* \le 3/2$ (see Lemma 2), $mT' < kT'/2 < \sum_{i=1}^n p_i$, and $T' \ge T$, so we have

$$\frac{\lceil\frac{k}{m}\rceil(T'+a)}{\lceil\frac{k^*}{m}\rceil(T+a)} \le \frac{\lceil\lceil\frac{3}{2}\rceil\frac{k^*}{m}\rceil}{\lceil\frac{k^*}{m}\rceil} \cdot \frac{T'+a}{T+a} \le \frac{2(T'+a)}{T+a}$$

and

$$\begin{aligned}\frac{\lceil\frac{k}{m}\rceil bT'}{b\sum_{i=1}^n p_i/m} &= \frac{\lceil\frac{k}{m}\rceil T'}{\sum_{i=1}^n p_i/m} \le \frac{\frac{k}{m}T' + T'}{\sum_{i=1}^n p_i/m} \\ &\le \frac{2\sum_{i=1}^n p_i/m + T'}{\sum_{i=1}^n p_i/m} \\ &= 2 + \frac{T'}{\sum_{i=1}^n p_i/m} < 3.\end{aligned}$$

Hence,

$$\frac{MC_{\max}^{\text{BFD-LPT}}}{MC_{\max}^*} \le \max\{2(T'+a)/(T+a), 3\}.$$

This completes the proof. □

Remark 1 Note that there is no polynomial time ρ-approximation algorithm for $Pm, MS[T,T], T_M(t) \equiv a||MC_{\max}$ for any $\rho < 2$, unless $P = NP$ (Xu et al. 2008), so it is easy to see that there is also no polynomial time ρ-approximation algorithm for our problem $Pm, MS[T,T'], T_M(t) = a + bt||MC_{\max}$ for any $\rho < 2$, unless $P = NP$.

4.2 Performance analysis of the FFD-LS algorithm for the single machine scheduling problem

We view each of the working interval between any two consecutive maintenance periods of a schedule as a bin. Before analyzing the FFD-LS algorithm, we first present some properties and lemmas.

Property 4 The optimal schedule of $1, MS[T,T], T_M(t) = a + bt, b \le 1||C_{\max}$ must have the minimum number of bins.

Proof Assume that the optimal schedule has k^* bins and there is a feasible schedule S with k bins. If possible, let $k < k^*$. Now, let the total processing times of the jobs in the last bin of the optimal schedule and the schedule S be x and y, respectively. It is easy to see that the makespan of the optimal schedule is

$$C_{\max}^* = (k^*-1)(T+a) + x + \left(\sum_{i=1}^n p_i - x\right)b,$$

and the makespan of the schedule S is

$$C_{\max}^S = (k-1)(T+a) + y + \left(\sum_{i=1}^n p_i - y\right)b.$$

Thus,

$$C_{\max}^* - C_{\max}^S = (k^*-k)(T+a) + (x-y)(1-b). \tag{1}$$

If $x \ge y$, then $C_{\max}^* - C_{\max}^S \ge (k^*-k)(T+a) > 0$, since $k^* > k$ and $b \le 1$; if $x < y$, then $C_{\max}^* - C_{\max}^S \ge (k^*-k)(T+a) + (x-y) \ge (T+a) + (x-y) > 0$, since $k^* > k$ and $y - x < T$. This implies that $C_{\max}^S < C_{\max}^*$, a contradiction. Therefore, the optimal schedule of $1, MS[T,T], T_M(t) = a + bt, b \le 1||C_{\max}$ must have the minimum number of bins. □

Lemma 5 (see Baase and Gelder 2000, p. 574) *If we pack the items by the FFD algorithm for the one dimensional bin-packing problem and $k > k^*$, where k is the number of bins obtained by the FFD algorithm and k^* is the optimal number of bins, then the size of each item in bins $B_{k^*+1}, B_{k^*+2}, \ldots, B_k$ is at most $W/3$.*

Although not formulated as a theorem or lemma using the bin-packing terminology in their paper, Ji et al. (2007), in fact, showed the following result.

Lemma 6 (Ji et al. 2007) *If we pack the items by the FFD algorithm for the one dimensional bin-packing problem and $k > k^*$, where k is the number of bins obtained by the FFD algorithm and k^* is the optimal number of bins, then we have*

(a) *if $k^* = 3$, then $k = 4$;*
(b) *if $k^* = 2$, then $k = 3$ and the total size of the items in the third bin is greater than $2W/3$.*

Now we give the worst-case bound of the FFD-LS algorithm.

Theorem 7 *For the problem $1, MS[T,T], T_M(t) = a + bt, b \le 1||C_{\max}$, the worst-case bound of the FFD-LS algorithm is 2.*

Proof Assume that the optimal schedule has k^* bins while the FFD-LS schedule has k bins. Without loss of generality, we assume that the "jobs" $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_{k-1}$ are processed according to the increasing numerical order of their indexes in the FFD-LS schedule, since the processing order of these jobs does not affect the makespan. Let the total processing times of the jobs in the last bin of the optimal schedule and the FFD-LS schedule be x and y, respectively. It is easy to see that the makespan of the optimal schedule is

$$\begin{aligned} C_{\max}^* &= (k^*-1)(T+a) + x + \left(\sum_{i=1}^n p_i - x\right)b \\ &= (k^*-1)(T+a) + (1-b)x + b\sum_{i=1}^n p_i, \end{aligned} \tag{2}$$

and the makespan of the FFD-LS schedule is

$$\begin{aligned} C_{\max}^{\text{FFD-LS}} &= (k-1)(T+a) + y + \left(\sum_{i=1}^n p_i - y\right)b \\ &= (k-1)(T+a) + (1-b)y + b\sum_{i=1}^n p_i. \end{aligned} \tag{3}$$

By (2), we have

$$k^* = 1 + \frac{C_{\max}^* - (1-b)x - b\sum_{i=1}^n p_i}{T+a}.$$

Note that $k \le 3k^*/2$ (see Lemma 2), then

$$k \le \frac{3}{2}\left(1 + \frac{C^*_{\max} - (1-b)x - b\sum_{i=1}^n p_i}{T+a}\right). \tag{4}$$

Substituting (4) into (3), we have

$$\begin{aligned} C^{\text{FFD-LS}}_{\max} &\le \left[\frac{3}{2}\left(1 + \frac{C^*_{\max} - (1-b)x - b\sum_{i=1}^n p_i}{T+a}\right) - 1\right] \\ &\quad \times (T+a) + (1-b)y + b\sum_{i=1}^n p_i \\ &= \frac{3}{2}C^*_{\max} + \frac{1}{2}(T+a) - \frac{3}{2}(1-b)x \\ &\quad + (1-b)y - \frac{b}{2}\sum_{i=1}^n p_i \\ &\le \frac{3}{2}C^*_{\max} + \frac{1}{2}(T+a) + (1-b)y \\ &\le \frac{3}{2}C^*_{\max} + \frac{1}{2}(T+a) + y. \end{aligned} \tag{5}$$

Note that $y \le T$, so we have

$$C^{\text{FFD-LS}}_{\max} \le \frac{3}{2}C^*_{\max} + \frac{1}{2}(3T+a). \tag{6}$$

If $k^* = 1$, it is clear that $C^{\text{FFD-LS}}_{\max} = C^*_{\max}$, and we are done. Thus, we assume in the following that $k^* > 1$. Now, if $k = k^*$, then by (2) and (3), we have $C^{\text{FFD-LS}}_{\max} = C^*_{\max} + (y-x)(1-b)$. Note that $(y-x)(1-b) \le |y-x| < T < C^*_{\max}$, so we have $C^{\text{FFD-LS}}_{\max} < 2C^*_{\max}$. If $k > k^*$, we consider the following three cases.

Case 1: $k^* \ge 4$. Thus, by (2), we have $C^*_{\max} \ge 3(T+a) \ge 3T+a$. Combining this with (6), we have $C^{\text{FFD-LS}}_{\max} \le 2C^*_{\max}$.

Case 2: $k^* = 3$. By Lemma 6, we have $k = 4$. Therefore, $C^{\text{FFD-LS}}_{\max} = 3(T+a) + (1-b)y + b\sum_{i=1}^n p_i \le 3(T+a) + y + b\sum_{i=1}^n p_i \le 4(T+a) + b\sum_{i=1}^n p_i$. On the other hand, $C^*_{\max} = 2(T+a) + (1-b)x + b\sum_{i=1}^n p_i \ge 2(T+a) + b\sum_{i=1}^n p_i$. So we have $C^{\text{FFD-LS}}_{\max} \le 2C^*_{\max}$.

Case 3: $k^* = 2$. By Lemma 5 and Lemma 6, we have $k = 4$, $y \le T/3$, and $x > 2T/3$. Therefore, $C^*_{\max} = (T+a) + (1-b)x + b\sum_{i=1}^n p_i \ge (T+a) + 2T(1-b)/3 + b\sum_{i=1}^n p_i$ and $C^{\text{FFD-LS}}_{\max} = 2(T+a) + (1-b)y + b\sum_{i=1}^n p_i \le 2(T+a) + T(1-b)/3 + b\sum_{i=1}^n p_i$. It follows that $C^{\text{FFD-LS}}_{\max} \le 2C^*_{\max}$.

Hence, we have completed the proof that the worst-case bound of the FFD-LS algorithm is not greater than 2. To show that this bound cannot be smaller than 2, consider the following instance. Let $T = 10$, $p_1 = p_2 = 4$, $p_3 = p_4 = p_5 = p_6 = 3$, $b = 1$, and a be an arbitrary integer. It is easy to see that $C^{\text{FFD-LS}}_{\max} = 40 + 2a$, while $C^*_{\max} = 30 + a$. It follows that $C^{\text{FFD-LS}}_{\max}/C^*_{\max} = (40+2a)/(30+a) \to 2$, as $a \to \infty$.

This completes the proof. □

Remark 2 Note that Ji et al. (2007) showed that there is no polynomial time approximation algorithm for the scheduling problem $1, MS[T,T], T_M(t) \equiv a||C_{\max}$ with a worst-case bound less than 2, unless $P = NP$. So we may conclude that there is also no polynomial time approximation algorithm for the scheduling problem $1, MS[T,T], T_M(t) = a + bt, b \le 1||C_{\max}$ with a worst-case bound less than 2, unless $P = NP$, and that the FFD-LS algorithm is the best possible polynomial time algorithm for $1, MS[T,T], T_M(t) = a + bt, b \le 1||C_{\max}$ if $P \ne NP$.

Remark 3 Property 4 is crucial for Theorem 7, since we can derive an estimate of the optimal makespan based on this property, and then determine the worst-case bound of the FFD-LS algorithm for the scheduling problem $1, MS[T,T], T_M(t) = a + bt, b \le 1||C_{\max}$. However, according to (1), it seems that it is not necessary for an optimal schedule of $1, MS[T,T], T_M(t) = a + bt, b > 2 + a/T||C_{\max}$ to have the minimum number of bins. Whether this is true may be an interesting problem for further study.

5 Conclusions

We consider two scheduling problems $1, MS[T,T], T_M(t) = a + bt, b \le 1||C_{\max}$ and $Pm, MS[T,T'], T_M(t) = a + bt||MC_{\max}$ in this paper. Both of the two problems are *NP*-hard, and there is no polynomial time algorithm for these problems with a worst-case bound less than 2, unless $P = NP$. We propose an approximation algorithm with worst-case bound at most $\max\{2(T'+a)/(T+a), 4\}$ for the parallel machine scheduling problem and a polynomial time 2-approximation algorithm for the single machine scheduling problem. Further research may focus on analyzing the worst-case bound of the Modified BFD-LPT for the parallel machine scheduling problem. It is also worth considering problems with other objectives or more practical maintenance requirements.

Acknowledgements This research was supported in part by the National 973 Fundamental Research Project of China (2002CB312200) and National Natural Science Foundation of China (60474023).

References

Baase, S., & Gelder, A. V. (2000). *Computer algorithms: introduction to design and analysis* (3rd ed.). Massachusetts: Addison-Wesley.

Coffman, Jr., E. G., Garey, M. R., & Johnson, D. S. (1978). An application of bin-packing to multiprocessor scheduling. *SIAM Journal on Computing*, *7*, 1–17.

Coffman, Jr., E. G., Garey, M. R., & Johnson, D. S. (1997). Approximation algorithms for bin packing: a survey. In D. S. Hochbaum (Ed.), *Approximation algorithms for NP-hard problems* (pp. 46–93). Boston: PWS.

Graham, R. L. (1966). Bounds for certain multiprocessor anomalies. *Bell System Technical Journal*, *45*, 1563–1581.

Graham, R. L., Lawer, E. L., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, *5*, 287–326.

Ji, M., He, Y., & Cheng, T. C. E. (2007). Single-machine scheduling with periodic maintenance to minimize makespan. *Computers & Operations Research*, *34*, 1764–1770.

Lee, C. Y. (2004). Machine scheduling with availability constraints. In J. Y.-T. Leung (Ed.), *Handbook of scheduling: algorithms, models, and performance analysis*. Boca Raton: Chapman & Hall/CRC (Chap. 22).

Sanlaville, E., & Schmidt, G. (1998). Machine scheduling with availability constraints. *Acta Informatica*, *35*, 795–811.

Schmidt, G. (2000). Scheduling with limited machine availability. *European Journal of Operational Research*, *121*, 1–15.

Simchi-Levi, D. (1994). New worst-case results for the bin packing problem. *Naval Research Logistics*, *41*, 579–585.

Xu, D. H., Sun, K. B., & Li, H. X. (2008). Parallel machine scheduling with almost periodic maintenance and non-preemptive jobs to minimize makespan. *Computers & Operations Research*, *35*, 1344–1349.