Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 547573, 8 pages
http://dx.doi.org/10.1155/2014/547573

Research Article

Best Possible Approximation Algorithms for Single Machine Scheduling with Increasing Linear Maintenance Durations

Xuefei Shi and Dehua Xu

School of Science, East China Institute of Technology, Nanchang, Jiangxi 330013, China

Correspondence should be addressed to Dehua Xu; dhxu@ecit.cn

Received 6 November 2013; Accepted 19 December 2013; Published 13 February 2014

Academic Editors: J. G. Barbosa and D. Oron

We consider a single machine scheduling problem with multiple maintenance activities, where the maintenance duration function is of the linear form $f(t) = a+bt$ with $a \geq 0$ and $b > 1$. We propose an approximation algorithm named FFD-LS2I with a worst-case bound of 2 for problem. We also show that there is no polynomial time approximation algorithm with a worst-case bound less than 2 for the problem with $b \geq 0$ unless $P = \text{NP}$, which implies that the FFD-LS2I algorithm is the best possible algorithm for the case $b > 1$ and that the FFD-LS algorithm, which is proposed in the literature, is the best possible algorithm for the case $b \leq 1$ both from the worst-case bound point of view.

1. Introduction

Scheduling with machine maintenance has attracted many researchers' attention in the last two decades (see, e.g., [1–13]), where high performance approximation algorithms are of great value not only theoretically but also practically.

In a recent paper, Xu et al. [14] study a single machine scheduling problem with multiple maintenance activities which can be formally described as follows. There are n independent jobs $J_1, J_2, \ldots, J_n$ to be processed on a single machine. The processing time of job J_i is p_i. All jobs are nonpreemptive and are available at time zero. The machine can process at most one job at a time. Assume that the machine has just finished a dummy maintenance activity at time zero. The length of the time interval between any two consecutive maintenance activities is T and prefixed. The amount of time needed to perform one maintenance activity is an increasing linear function $T_M(t) = a + bt$ of the total processing time t of the jobs that are processed after its latest previous maintenance activity, where a and b are prefixed nonnegative real numbers with $b \leq 1$. The objective is to schedule all the jobs to the machine such that the makespan, that is, the completion time of the last finished job, is minimized. Extending the well-known three-field $\alpha|\beta|\gamma$ classification scheme suggested by Graham et al. [15], the problem is denoted by $1, MS[T,T], T_M(t) = a + bt, b \leq 1 \parallel C_{\max}$ in Xu et al. [14]. For simplicity, in what follows, we denote this problem by $P_{(b\leq 1)}$ and its counterpart where $b > 1$ by $P_{(b>1)}$. Additionally, the problem is denoted by $P_{(b>0)}$ if $b > 0$.

Xu et al. [14] first propose an approximation algorithm named FFD-LS for the problem $P_{(b\leq 1)}$ and then show that its worst-case bound is 2. The underlying idea of their FFD-LS algorithm is straightforward. Viewing the jobs as items, it first packs these items into some bins with the same capacity of T by the classical bin-packing algorithm FFD. Let the "jobs" in each used bin plus a maintenance activity whose duration is computed according to the maintenance function $T_M(t)$ be viewed as a hybrid job. Assign all the hybrid jobs except the last one (i.e., the one with the largest index) to the machine by the classical LS algorithm. Assign the last hybrid jobs to the machine.

For the sake of convenience and completeness, the bin-packing problem, the FFD algorithm, the LS algorithm, and the FFD-LS algorithm are presented as follows.

Bin-Packing Problem (see, e.g., Coffman et al. [16]). 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 $\sum_{a_i \in B_j} s(a_i) \le W, 1 \le j \le m$).

Algorithm FFD (see, e.g., Coffman et al. [16]). Sort all the items such that $s(a_1) \ge s(a_2) \ge \cdots \ge s(a_n)$; for $i = 1, 2, \ldots, n$, item a_i is packed in the first (lowest-indexed) bin into which it will fit; that is, if there is any partially filled bin B_j with $\text{level}(B_j) \le W - s(a_i)$ where $\text{level}(B_j)$ is the sum of the sizes of the items that have been packed into bin B_j, 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 LS (see, e.g., Pinedo [17]). Put all the jobs on a list in arbitrary order; then process the jobs consecutively as early as possible.

Algorithm FFD-LS (see Xu et al. [14])

Step 1. Construct a bin-packing instance 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. Using the FFD algorithm, assume that 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 with a processing time of $p_j^{(i)}$ 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 $a + \text{level}(B_i)b$.

Step 3. Let $\mathcal{J}_i$ be viewed as a single job with the processing time of $T + a + \text{level}(B_i)b$. 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.

Although the FFD-LS algorithm is originally proposed for $P_{(b\le 1)}$, it can also be applied to $P_{(b>1)}$ without any modification. However, as can be seen in the next section, there exists an example showing that the worst-case bound of the FFD-LS algorithm can be arbitrarily large for $P_{(b>1)}$. As a result, we then focus our attention on studying its worst-case bound for $P_{(b>1)}$ in detail. Specifically, we think of each interval between two consecutive maintenance activities as a batch with a capacity T and show that its worst-case bound is 2 for $P_{(b>1)}$ if the number of nonempty batches in the FFD-LS schedule is not 2. In order to avoid the arbitrarily large worst-case bound, in Section 3, we modify the FFD-LS algorithm and get a new approximation algorithm named FFD-LS2I algorithm with a worst-case bound of 2 for $P_{(b>1)}$ by adding a step to the FFD-LS algorithm. In Section 4, we study the nonapproximability of the problem and show that there is no polynomial time approximation algorithm with a worst-case bound less than 2 for the problem $P_{(b>0)}$, which implies that the FFD-LS2I algorithm is the best possible algorithm for $P_{(b>1)}$ and that the FFD-LS is the best possible algorithm for $P_{(b\le 1)}$ both from the worst-case bound point of view. Finally, in Section 5, we give an answer to another open problem proposed in Xu et al. [14].

2. Worst-Case Bound of the FFD-LS Algorithm for $P_{(b>1)}$

Example 1. Consider the scheduling problem $P_{(b>1)}$ with the following instance: $n = 2, T = b, p_1 = b$, and $p_2 = 1$. It is easy to see that the FFD-LS algorithm assigns job J_1 to the first batch and J_2 to the second batch and that the corresponding makespan is $C_{\max}^{\text{FFD-LS}} = b + a + b^2 + 1 = b^2 + b + a + 1$. Note that the optimal schedule assigns job J_2 to the first batch and J_1 to the second batch and that the optimal makespan is $C_{\max}^* = 3b + a$. Clearly, $C_{\max}^{\text{FFD-LS}}/C_{\max}^* \to \infty$ as $b \to \infty$. In other words, the worst-case bound of the FFD-LS algorithm can be arbitrarily large.

Now, let $\Lambda = \sum_{i=1}^{n} p_i$. Let S^* be an optimal schedule with k^* nonempty batches $\mathcal{B}_1^*, \ldots, \mathcal{B}_{k^*}^*$ which are indexed according to the processing order. Recall that we assume that there are totally k bins being used by the FFD algorithm in the first step of the FFD-LS algorithm. So, for consistency, we may assume that there are totally k nonempty batches in the FFD-LS schedule. For convenience, in what follows, these batches are denoted by $\mathcal{B}_1, \ldots, \mathcal{B}_k$ which are also indexed according to the processing order. Let the total processing times of the jobs in batch $\mathcal{B}_i^*$ and batch $\mathcal{B}_i$ be denoted by λ_i^* and λ_i, respectively. It is easy to see that the makespan of the optimal schedule is

$$C_{\max}^* = (k^* - 1)(T + a) + b(\Lambda - \lambda_{k^*}^*) + \lambda_{k^*}^* \quad (1)$$

and that the makespan of the FFD-LS schedule is

$$C_{\max}^{\text{FFD-LS}} = (k - 1)(T + a) + b(\Lambda - \lambda_k) + \lambda_k. \quad (2)$$

In what follows, we shall study the worst-case bound of the FFD-LS algorithm for $P_{(b>1)}$ in detail, which can help us develop a more sophisticated approximation algorithm. Before the study of the worst-case bound, we first present some lemmas.

Lemma 2. *For the problem $P_{(b>1)}$, the total processing time of the jobs in any two nonempty batches is strictly larger than T in S^*.*

Proof (by contradiction). It suffices to show that $\lambda_i^* + \lambda_j^* > T$ for $1 \le i < j \le k^*$.

If there exist two batches, say $\mathcal{B}_i^*$ and $\mathcal{B}_j^*$, such that $\lambda_i^* + \lambda_j^* \le T$, where $1 \le i < j \le k^*$, consider the following two cases.

Case 1 ($1 \le i < j < k^*$). Note that $\lambda_i^* + \lambda_j^* \le T$. Now, reschedule all the jobs of $\mathcal{B}_i^*$ into $\mathcal{B}_j^*$ and delete $\mathcal{B}_i^*$ with the corresponding maintenance activity. Let the new schedule be denoted by S'. It is easy to see that S' is a feasible schedule with a makespan of

$$C'_{\max} = (k^* - 2)(T + a) + b(\Lambda - \lambda_{k^*}^*) + \lambda_{k^*}^*. \quad (3)$$

Comparing (1) with (3), we have $C'_{\max} < C_{\max}^*$, which contradicts the optimality of S^*.

Case 2 ($1 \le i < k^*$ and $j = k^*$). Similar to Case 1, reschedule all the jobs of $\mathcal{B}_i^*$ into $\mathcal{B}_j^*$ and delete $\mathcal{B}_i^*$ with

its corresponding maintenance. Let the new schedule be denoted by S''. It is easy to see that S'' is a feasible schedule with a makespan of

$$C''_{\max} = (k^* - 2)(T + a) + b(\Lambda - \lambda^*_{k^*} - \lambda^*_i) + (\lambda^*_{k^*} + \lambda^*_i). \tag{4}$$

Recall that $b > 1$. Comparing (1) with (4), we have $C''_{\max} < C^*_{\max}$, which again contradicts the optimality of S^*.

This completes the proof. □

Lemma 3. *For problem $P_{(b>1)}$, let S' be a feasible schedule with two nonempty batches and let S'' be a feasible schedule with three nonempty batches. Let the total processing times of the jobs in the first batch of S' and in the second batch of S' be denoted by λ'_1 and λ'_2, respectively. Let the total processing times of the jobs in the first batch of S'', in the second batch of S'', and in the third batch of S'' be denoted by λ''_1, λ''_2 and λ''_3, respectively. If $\lambda'_1 + \lambda'_2 > T$ and $\lambda''_i + \lambda''_j > T$ for $1 \le i < j \le 3$, then one has $C'_{\max} < C''_{\max}$, where $C'_{\max}$ and $C''_{\max}$ are the makespans of S' and S'', respectively.*

Proof. It is easy to see that

$$C'_{\max} = T + a + b\lambda'_1 + \lambda'_2, \tag{5}$$

$$C''_{\max} = 2(T + a) + b(\lambda''_1 + \lambda''_2) + \lambda''_3. \tag{6}$$

Combining (5) with (6), we have

$$C'_{\max} = C''_{\max} + (\lambda'_2 - T) - a + b(\lambda'_1 - \lambda''_1 - \lambda''_2) - \lambda''_3. \tag{7}$$

Recall that $\lambda''_1 + \lambda''_2 > T$. So we have

$$C'_{\max} < C''_{\max} + (\lambda'_2 - T) - a + b(\lambda'_1 - T) - \lambda''_3. \tag{8}$$

Note that $\lambda'_1 \le T$ and $\lambda'_2 \le T$. So we have

$$C'_{\max} < C''_{\max}. \tag{9}$$

This completes the proof. □

Lemma 4. $k \le 3k^*/2$.

Proof. View each interval between two consecutive maintenance activities as a bin with capacity T and the jobs J_i as items a_i with a size of p_i. Let k' be the minimum number of nonempty bins that is needed to pack all the n items. It is well known that $k \le 3k'/2$ (see Simchi-Levi [18]). Note that $k' \le k^*$. So we have $k \le 3k^*/2$. This completes the proof. □

Lemma 5. *For the problem $P_{(b>1)}$, if $T < \Lambda \le 2T$, then $2 \le k^* \le 3$ and $2 \le k \le 3$.*

Proof. By $T < \Lambda$, it is easy to see that $2 \le k^*$ and $2 \le k$ since that one batch is clearly not sufficient in any feasible schedule. On the other hand, by Lemma 2, we have $k^* \le 3$ for otherwise we have $\Lambda > 2T$, a contradiction. Note that the total processing time of the jobs in any two batches in the FFD-LS schedule is strictly larger than T. We thus have $k \le 3$. To sum up, we have $2 \le k^* \le 3$ and $2 \le k \le 3$. This completes the proof. □

Now, we are ready to evaluate the worst-case bound of the FFD-LS algorithm in detail.

Theorem 6. *For the problem $P_{(b>1)}$, the worst-case bound of the FFD-LS algorithm is not greater than 2 if $\Lambda > 2T$.*

Proof. By (1), we have

$$k^* = 1 + \frac{C^*_{\max} + (b-1)\lambda^*_{k^*} - b\Lambda}{T + a}. \tag{10}$$

Note that $k \le 3k^*/2$ (see Lemma 4). So we have

$$k \le \frac{3}{2}\left(1 + \frac{C^*_{\max} + (b-1)\lambda^*_{k^*} - b\Lambda}{T + a}\right). \tag{11}$$

Substituting (11) into (2), we have

$$C^{\text{FFD-LS}}_{\max} \le \frac{3}{2}C^*_{\max} + \frac{1}{2}(T + a) + \frac{3}{2}(b-1)\lambda^*_{k^*} - (b-1)\lambda_k - \frac{b}{2}\Lambda. \tag{12}$$

Now, consider the following two cases.

Case 1 ($\Lambda \ge 3T$). In this case, we have

$$\frac{b}{2}\Lambda \ge \frac{b}{2}3T > \frac{3}{2}(b-1)T. \tag{13}$$

Substituting (13) into (12), we have

$$C^{\text{FFD-LS}}_{\max} \le \frac{3}{2}C^*_{\max} + \frac{1}{2}(T + a) + \frac{3}{2}(b-1)(\lambda^*_{k^*} - T) - (b-1)\lambda_k. \tag{14}$$

Note that $\lambda^*_{k^*} \le T$, $-(b-1)\lambda_k \le 0$, and $T + a < C^*_{\max}$. Combining these three inequalities with (14), we have $C^{\text{FFD-LS}}_{\max} < 2C^*_{\max}$ and we are done.

Case 2 ($2T < \Lambda < 3T$). In this case, it is easy to see that $k^* \ge 3$ because two batches are obviously not sufficient in any optimal solution. Note that the total processing time of the jobs in any two nonempty batches is strictly larger than T in the FFD-LS schedule. We thus have $k \le 5$ for otherwise the total processing times of the jobs will be strictly larger than $3T$, which violates the assumption of this case.

Combining $k^* \ge 3$ with (1), we have

$$C^*_{\max} \ge 2(T + a) + b(\Lambda - \lambda^*_{k^*}) + \lambda^*_{k^*}. \tag{15}$$

By simple algebra, we have

$$C^*_{\max} \ge 2(T + a) + b\Lambda - (b-1)\lambda^*_{k^*}. \tag{16}$$

Note that $0 < \lambda^*_{k^*} \le T$. So we have

$$C^*_{\max} \ge 2(T + a) + b\Lambda - (b-1)T. \tag{17}$$

Similarly, combining $k \le 5$, $0 < \lambda_k \le T$, and (2), we have

$$C_{\max}^{\text{FFD-LS}} < 4(T + a) + b\Lambda. \tag{18}$$

By simple algebra, (18) becomes

$$C_{\max}^{\text{FFD-LS}} < 2(2(T + a) + b\Lambda - (b - 1)T) - b(\Lambda - 2T) - 2T. \tag{19}$$

Substituting (17) into (19), we have

$$C_{\max}^{\text{FFD-LS}} < 2C_{\max}^{*} - b(\Lambda - 2T) - 2T. \tag{20}$$

Recall that $2T < \Lambda$; we thus have $C_{\max}^{\text{FFD-LS}} < 2C_{\max}^{*}$.

This completes the proof. □

Theorem 7. *For the problem $P_{(b>1)}$, the worst-case bound of the FFD-LS algorithm is not greater than 2 if $T < \Lambda \le 2T$ and $k = 3$.*

Proof. Substituting $k = 3$ into (2), we have

$$C_{\max}^{\text{FFD-LS}} = 2(T + a) + b\Lambda - (b - 1)\lambda_3. \tag{21}$$

By Lemma 5, we have $2 \le k^{*} \le 3$. Now, consider the following two cases.

Case 1 ($k^{*} = 2$). Substituting $k^{*} = 2$ into (1), we have

$$C_{\max}^{*} \ge T + a + b(\Lambda - \lambda_2^{*}) + \lambda_2^{*}. \tag{22}$$

Combining (22) with (21), we have

$$C_{\max}^{\text{FFD-LS}} = 2C_{\max}^{*} - (b - 1)(\Lambda + \lambda_3 - 2\lambda_2^{*}) - \Lambda. \tag{23}$$

Note that $\Lambda = \lambda_1 + \lambda_2 + \lambda_3$. Combining this with (23), we have

$$C_{\max}^{\text{FFD-LS}} \le 2C_{\max}^{*} - (b - 1)((\lambda_1 + \lambda_3 - \lambda_2^{*}) + (\lambda_2 + \lambda_3 - \lambda_2^{*})) - \Lambda. \tag{24}$$

Note that $\lambda_1 + \lambda_3 > T \ge \lambda_2^{*}$ and that $\lambda_2 + \lambda_3 > T \ge \lambda_2^{*}$. So, by (24), we have $C_{\max}^{\text{FFD-LS}} \le 2C_{\max}^{*}$.

Case 2 ($k^{*} = 3$). Substituting $k^{*} = 3$ into (1), we have

$$C_{\max}^{*} = T + a + b(\Lambda - \lambda_3^{*}) + \lambda_3^{*}. \tag{25}$$

Combining (25) with (21), we have

$$C_{\max}^{\text{FFD-LS}} = C_{\max}^{*} + (b - 1)(\lambda_3^{*} - \lambda_3). \tag{26}$$

Note that $\lambda_3^{*} - \lambda_3 \le T$ and $b - 1 \le b$. So we have

$$C_{\max}^{\text{FFD-LS}} \le C_{\max}^{*} + bT. \tag{27}$$

Note that $\Lambda - \lambda_3^{*} = \lambda_1^{*} + \lambda_2^{*} > T$. Combining this with (25), we have $C_{\max}^{*} > bT$. Substituting this into (27), we have $C_{\max}^{\text{FFD-LS}} < 2C_{\max}^{*}$.

This completes the proof. □

Theorem 8. *For the problem $P_{(b>1)}$, the worst-case bound of the FFD-LS algorithm is not greater than 2 if $k \ne 2$.*

Proof. Consider the following three cases.

Case 1 ($k = 1$). It is trivially obvious that in this case we have $C_{\max}^{\text{FFD-LS}} = C_{\max}^{*}$, and we are done.

Case 2 ($k = 3$). Clearly, in this case, we have $\Lambda > T$. Now, if $T < \Lambda \le 2T$, then, by Theorem 7, we know that the worst-case bound of the FFD-LS algorithm is not greater than 2, and we are done. If $2T < \Lambda$, then by Theorem 6, we know that the worst-case bound of the FFD-LS algorithm is not greater than 2 either, and we are also done. This complete the proof of Case 2.

Case 3 ($k \ge 4$). Clearly, in this case, we have $\Lambda > 2T$. Again, by Theorem 6, we know that the worst-case bound of the FFD-LS algorithm is not greater than 2, and we are done.

This completes the proof. □

3. A Modified Approximation Algorithm and Its Worst-Case Bound for $P_{(b>1)}$

We know from Example 1 and Theorem 8 that the worst-case bound of the FFD-LS algorithm can be arbitrarily large if $k = 2$ and is 2 otherwise. Following the notation in Xu et al. [14], $\mathcal{J}_i$ is the hybrid job which consists of the ith nonempty bin obtained by the FFD algorithm and the corresponding maintenance activity. Note that $\mathcal{J}_1$ is scheduled before $\mathcal{J}_2$ in the FFD-LS schedule. So we have $\lambda_1 = \text{level}(B_1)$ and $\lambda_2 = \text{level}(B_2)$. In order to avoid the arbitrarily large worst-case bound, as we observe, we only need to add the following step after the FFD-LS algorithm.

Step 4. If $k = 2$ and $\text{level}(B_2) < \text{level}(B_1)$, interchange $\mathcal{J}_1$ with $\mathcal{J}_2$.

Let us call this modified algorithm FFD-LS2I, which applies algorithm FFD-LS first and then applies Step 4. Clearly, the computational complexity of the FFD-LS2I is $O(n^2)$. Let the FFD-LS2I schedule be denoted by $S^{\star}$ and let the total processing times of the jobs in the ith batch B_i^{*} of $S^{\star}$ be denoted by λ_i^{*}. It is easy to see that the number of the nonempty batches in $S^{\star}$ is also k because the fourth step of the FFD-LS2I algorithm does not change the number of batches.

In order to show that the worst-case bound of the FFD-LS2I algorithm is 2, by the structure of the FFD-LS2I algorithm and Theorem 8, we only need to consider the case $k = 2$.

Theorem 9. *For the problem $P_{(b>1)}$, the worst-case bound of the FFD-LS2I algorithm is not greater than 2 if $k = 2$.*

Proof. Clearly, $k = 2$ implies $T < \Lambda \le 2T$. Hence, by Lemma 4, we have $2 \le k^{*} \le 3$. Note that the FFD-LS2I schedule has two nonempty batches and that Lemma 3 indicates that the makespan of a feasible schedule with two batches is always less than that of a feasible schedule with three batches. So we have $k^{*} = 2$.

Substituting $k^* = 2$ into (1), we have

$$C^*_{\max} = T + a + b\left(\Lambda - \lambda^*_2\right) + \lambda^*_2. \tag{28}$$

Similar to (2), the makespan of the FFD-LS2I schedule is

$$C^{\text{FFD-LS2I}}_{\max} = T + a + b\left(\Lambda - \lambda^\star_2\right) + \lambda^\star_2. \tag{29}$$

Combining these two equations, we have

$$C^{\text{FFD-LS2I}}_{\max} = 2C^*_{\max} - T - a - b\left(\Lambda + \lambda^\star_2 - 2\lambda^*_2\right) + \left(\lambda^\star_2 - 2\lambda^*_2\right). \tag{30}$$

Note that $\lambda^*_1 + \lambda^*_2 = \Lambda > T$. We claim that $\lambda^*_2 > T/2$ for otherwise we have $\lambda^*_1 > T/2$ and thus we can get a schedule with a smaller makespan by interchanging all the jobs in the first batch of S^* with those in the second batch of S^*, which clearly violates the optimality of S^*. Combining this with $\lambda^\star_2 \le T$, we have

$$\lambda^\star_2 - 2\lambda^*_2 \le \lambda^\star_2 - T \le 0. \tag{31}$$

Case 1 ($4T/3 \le \Lambda \le 2T$). Note that $\lambda^\star_1 \le \lambda^\star_2$ and that $\lambda^\star_1 + \lambda^\star_2 = \Lambda$. So we have $\lambda^\star_2 \ge 2T/3$. And thus $\Lambda + \lambda^\star_2 - 2\lambda^*_2 \ge 2T - 2\lambda^*_2$. Combining this with $\lambda^*_2 \le T$, we have

$$\Lambda + \lambda^\star_2 - 2\lambda^*_2 \ge 0. \tag{32}$$

Substituting (31) and (32) into (30), we have

$$C^{\text{FFD-LS2I}}_{\max} \le 2C^*_{\max} - T - a, \tag{33}$$

which implies that $C^{\text{FFD-LS2I}}_{\max} < 2C^*_{\max}$.

Case 2 ($T < \Lambda < 4T/3$). Consider the schedule $S^\diamond$ which schedules the hybrid job $\mathcal{J}_2$ first and then the hybrid job $\mathcal{J}_1$. Let the total processing time of the jobs in the first batch of $S^\diamond$ and in the second batch of $S^\diamond$ be denoted by $\lambda^\diamond_1$ and $\lambda^\diamond_2$, respectively. Clearly, $\lambda^\diamond_1 + \lambda^\diamond_2 = \Lambda$. Note that Step 4 schedules the hybrid job with the highest level last. So we have $\lambda^\diamond_2 \le \lambda^\star_2$.

Let the makespan of the schedule $S^\diamond$ be denoted by $C^\diamond_{\max}$. It is easy to see that

$$C^\diamond_{\max} = T + a + b\left(\Lambda - \lambda^\diamond_2\right) + \lambda^\diamond_2. \tag{34}$$

Recall that $\lambda^\diamond_2 \le \lambda^\star_2$ and $b > 1$. Comparing (29) with (34), we have

$$C^{\text{FFD-LS2I}}_{\max} \le C^\diamond_{\max}. \tag{35}$$

Now, consider two subcases.

Subcase 2.1. There is only one job, say job J_u, in the first batch of the schedule $S^\diamond$. Without loss of generality, we assume that the jobs in $\mathcal{J}_1$ are ordered in nondecreasing order of their processing times; that is, $p^{(1)}_1 \ge p^{(1)}_2 \ge \cdots \ge p^{(1)}_{k_1}$. Suppose that $p^{(1)}_s \ge p_u > p^{(1)}_{s+1}$. Clearly, we have

$$\sum_{i=1}^{s} p^{(1)}_i + p_u > T, \tag{36}$$

$$\lambda^\diamond_2 = \Lambda - p_u. \tag{37}$$

Now, consider the position of job J_u in the optimal schedule S^*. If J_u is in the first batch of S^*, then we have $\lambda^*_2 \le \Lambda - p_u$. Combine this with (37), we have $\lambda^\diamond_2 \ge \lambda^*_2$. If J_u is in the second batch of S^*, then at least one job, say $J^{(1)}_v$ with $1 \le v \le s$, must be assigned to the first batch of S^*. To see this is the case, suppose all the jobs $J^{(1)}_1, \ldots, J^{(1)}_s$ are assigned to the second batch of S^*. We thus have $\lambda^*_2 \ge \sum_{i=1}^{s} p^{(1)}_i + p_u$. Combining this with (36), we have $\lambda^*_2 > T$, which implies that S^* is infeasible, clearly, a contradiction. Now, suppose $J^{(1)}_v$ is in the first batch of S^*, where $1 \le v \le s$. Note that $p^{(1)}_v \ge p_u$. So we have $\lambda^*_2 \le \Lambda - p^{(1)}_v \le \Lambda - p_u$. Combining this with (37), we have $\lambda^\diamond_2 \ge \lambda^*_2$. So, in either case, we have $\lambda^\diamond_2 \ge \lambda^*_2$. Combining this with (28) and (34), we have $C^\diamond_{\max} \le C^*_{\max}$. Combining this with (35), we have $C^{\text{FFD-LS2I}}_{\max} \le C^*_{\max}$, which implies that $C^{\text{FFD-LS2I}}_{\max} = C^*_{\max}$ by the optimality of S^*.

Subcase 2.2. There are at least two jobs in the first batch of the schedule $S^\diamond$. Let J_l be the job with the smallest processing time in the first batch of the schedule $S^\diamond$. Clearly, we have

$$\begin{aligned} \lambda^\diamond_1 &\ge 2p_l, \\ \lambda^\diamond_2 + p_l &> T \ge \lambda^*_2. \end{aligned} \tag{38}$$

Combining (28) with (34), we have

$$C^\diamond_{\max} = 2C^*_{\max} - T - a - b\left(\Lambda + \lambda^\diamond_2 - 2\lambda^*_2\right) + \left(\lambda^\diamond_2 - 2\lambda^*_2\right). \tag{39}$$

By (38), we know that

$$\Lambda + \lambda^\diamond_2 - 2\lambda^*_2 \ge \Lambda - \lambda^\diamond_2 - 2p_l. \tag{40}$$

Recall that $\Lambda - \lambda^\diamond_2 = \lambda^\diamond_1$ and that $\lambda^\diamond_1 \ge 2p_l$. So we have

$$\Lambda + \lambda^\diamond_2 - 2\lambda^*_2 \ge 0. \tag{41}$$

By a similar reasoning as in the arguments of deducing (31), we have

$$\lambda^\diamond_2 - 2\lambda^*_2 \le \lambda^\diamond_2 - T \le 0. \tag{42}$$

Substituting (41) and (42) into (39), we have

$$C^\diamond_{\max} \le 2C^*_{\max} - T - a, \tag{43}$$

which implies that $C^\diamond_{\max} \le 2C^*_{\max}$. Combining this with (35), we have $C^{\text{FFD-LS2I}}_{\max} \le 2C^*_{\max}$.

This completes the proof. □

Now, we are ready to present the worst-case bound of the FFD-LS2I algorithm.

Theorem 10. *For the problem $P_{(b>1)}$, the worst-case bound of the FFD-LS2I algorithm is 2 and the bound is tight.*

Proof. If $k \ne 2$, it is clear that the FFD-LS2I algorithm is reduced to the FFD-LS algorithm, and thus, by Theorem 8, we know that the worst-case bound of the FFD-LS2I algorithm

is not greater than 2. If $k = 2$, then, by Theorem 9, we know that the worst-case bound of the FFD-LS2I algorithm is not greater than 2 either. Hence, we have completed the proof that the worst-case bound of the FFD-LS2I algorithm is not greater than 2.

To show that this bound cannot be smaller than 2, consider the following instance. Let $T = 12$, $p_1 = 8 - 2/b$, $p_2 = p_3 = 4+1/b$, $p_4 = 4$, $p_5 = 4-2/b$, $p_6 = 2/b$, and $a = 1$. It is easy to see that $C_{\max}^{\text{FFD-LS2I}} = 24 + 24b + 2/b$ while $C_{\max}^{*} = 25 + 12b$. It follows that $C_{\max}^{\text{FFD-LS2I}}/C_{\max}^{*} = (24 + 24b)/(25 + 12b) \to 2$ as $b \to \infty$.

This completes the proof. □

4. Nonapproximability

In this section, we shall show that it is impossible to have a polynomial time approximation algorithm with a worst-case bound of less than 2 for the scheduling problem $P_{(b\geq 0)}$ unless $P = \text{NP}$, which implies that the proposed algorithm FFD-LS2I is the best possible approximation algorithm for $P_{(b>1)}$ from the worst-case bound point of view and that the FFD-LS algorithm proposed by Xu et al. [14] is the best possible approximation algorithm for $P_{(b\leq 1)}$ from the same point of view.

Note that the case $b = 0$ has been considered in Ji et al. [3] (see the following lemma, that is, Lemma 11), so we only need to consider the case $b > 0$, that is, the problem $P_{(b>0)}$.

Lemma 11 (Ji et al. [3]). *There is no polynomial time approximation algorithm with a worst-case bound less than 2 for the scheduling problem $P_{(b=0)}$ unless $P = NP$.*

The underlying idea of our approach for problem $P_{(b>0)}$ is by contradiction. Specifically, we shall show that if there exists an approximation algorithm with a worst-case bound of $2-\epsilon$ with $0 < \epsilon < 1$ for $P_{(b>0)}$ then it can be used to establish a polynomial time algorithm for the well-known NP-complete problem Partition. This clearly leads to a contradiction if $P \neq \text{NP}$. Hence, such an approximation algorithm for the scheduling problem $P_{(b>0)}$ cannot exist unless $P = \text{NP}$.

Partition Problem (see, e.g., Garey and Johnson [19]). Given n positive integers $h_1, h_2, \ldots, h_n$ with $\sum_{i=1}^{n} h_i = 2H$, does there exist a set $X \subseteq \{1, 2, \ldots, n\}$ with $\sum_{i\in X} h_i = H$?

For any fixed positive number $\epsilon < 1$ and an instance I of the Partition problem, we construct an instance II of the scheduling problem $P_{(b>0)}$ as follows. There are n jobs $J_1, J_2, \ldots, J_n$ and the processing time of job J_i is $p_i = h_i$. Let $T = H$ and $a = (2T + bT)(1 - \epsilon)/\epsilon + 1$. It is clear that this construction can be performed in polynomial time.

Theorem 12. *If there exists a polynomial time approximation algorithm A_ϵ with a worst-case bound of $2-\epsilon$ for some positive number $\epsilon < 1$ for the scheduling problem $P_{(b>0)}$, then there exists a polynomial time algorithm for the Partition problem.*

Proof. Given any instance I of the Partition problem, we construct the corresponding instance II of the scheduling problem $P_{(b>0)}$ in polynomial time as stated above. Let $C_{\max}^{A_\epsilon}$ and $C_{\max}^{*}$ be the makespan of the schedule produced by the approximation algorithm A_ϵ and the makespan of an optimal schedule S^* for the instance II of $P_{(b>0)}$, respectively. We shall show that the instance I of the Partition problem can be answered by merely comparing the values of $C_{\max}^{A_\epsilon}$ and $2(T + a) + bT$.

To see that this is the case, let us apply algorithm A_ϵ to instance II. We claim that if $C_{\max}^{A_\epsilon} \leq 2(T + a) + bT$ then there exists a solution to instance I of the Partition problem and, otherwise, there does not exist a solution to instance I. In what follows, we shall show that this claim is correct.

Case 1 ($C_{\max}^{A_\epsilon} \leq 2(T+a)+bT$). Clearly, we have $C_{\max}^{*} \leq 2(T + a) + bT$ because $C_{\max}^{*} \leq C_{\max}^{A_\epsilon}$. Let k^* be the total number of batches used in S^*. Note that $T = H$ and $\Lambda = \sum_{i=1}^{n} p_i = 2H$, so we have $k^* \geq 2$.

Let λ_i^* be the total processing time of the jobs in the ith batch of S^*. If $k^* \geq 3$, then, by (1), we have $C_{\max}^{*} \geq 2(T + a) + b(\Lambda - \lambda_{k^*}^{*}) + \lambda_{k^*}^{*}$. Note that $\Lambda - \lambda_{k^*}^{*} = \sum_{i=1}^{k^*-1} \lambda_i^{*} \geq \lambda_1^{*} + \lambda_2^{*} > T$. So we have $C_{\max}^{*} > 2(T + a) + bT$, which contradicts the inequality $C_{\max}^{*} \leq 2(T+a)+bT$. This implies that $k^* = 2$. That is, the optimal schedule S^* has two nonempty batches. Now, let X be the set of indices of the jobs scheduled in the first batch of the optimal schedules. Clearly we have $\sum_{i\in X} p_i = T$. Recall that $T = H$ and $p_i = h_i$ for $i = 1, 2, \ldots, n$. So we have $\sum_{i\in X} h_i = H$, which implies that there exists a solution to the instance I of the Partition problem.

Case 2 ($C_{\max}^{A_\epsilon} > 2(T + a) + bT$). Recall that the worst-case bound of the algorithm A_ϵ is $2 - \epsilon$. So we have

$$C_{\max}^{*} \geq \frac{C_{A_\epsilon}}{2-\epsilon} > \frac{2(T + a) + bT}{2-\epsilon}. \tag{44}$$

Recall that $a = (2T + bT)(1 - \epsilon)/\epsilon + 1$. So we have

$$\epsilon = \frac{2T + bT}{2T + bT + a - 1}. \tag{45}$$

Substituting (45) into (44), we have

$$\begin{aligned} C_{\max}^{*} &> \frac{2(T + a) + bT}{2 - (2T + bT)/(2T + bT + a - 1)} \\ &= 2T + bT + a + \frac{2T + bT}{2T + bT + 2a - 2} \geq 2T + bT + a. \end{aligned} \tag{46}$$

If there exists a solution, namely, set X, to the instance I of the Partition problem, let the jobs whose indices belong to the set X be scheduled into the first batch and the remaining jobs the second batch; we then obtain a schedule with a makespan of $2T + a + bT$, which clearly contradicts (46). Hence, there does not exist a solution to the instance I of the Partition problem.

This completes the proof. □

By Theorem 10 and the fact that no NP-complete problem can be solved by a polynomial time algorithm unless $P = \text{NP}$, we have the following result.

Theorem 13. *There is no polynomial time approximation algorithm with a worst-case bound less than 2 for the scheduling problem $P_{(b>0)}$ unless $P = NP$.*

Combining Lemma 11 and Theorem 13, we have the following result.

Theorem 14. *There is no polynomial time approximation algorithm with a worst-case bound less than 2 for the scheduling problem $P_{(b\geq 0)}$ unless $P = NP$.*

Recall that we have showed in Theorem 10 that the worst-case bound of the FFD-LS2I algorithm is 2 for $P_{(b>1)}$. Combining this with Theorem 14, we have the following result.

Theorem 15. *The FFD-LS2I algorithm is the best possible polynomial time approximation algorithm for $P_{(b>1)}$ unless $P = NP$ from the worst-case bound point of view.*

Recall that Xu et al. [14] have showed that the worst-case bound of the FFD-LS algorithm is 2 for $P_{(b\leq 1)}$. Combining this with Theorem 14, we have the following result.

Theorem 16. *The FFD-LS algorithm is the best possible polynomial time approximation algorithm for $P_{(b\leq 1)}$ unless $P = NP$ from the worst-case bound point of view.*

5. An Answer to Another Open Problem

Viewing each batch as a bin, in Remark 3 of Section 4 of the paper of Xu et al. [14], Xu et al. claim that "it seems that it is not necessary for an optimal schedule of 1, $MS[T, T], T_M(t) = a + bt, b > 2 + a/T \parallel C_{\max}$ to have the minimum number of bins. Whether this is true may be an interesting problem for further study." The following example gives an answer to this open problem.

Example 17. Consider the following instance for the scheduling problem under consideration: $T = 20$, $a = 1$, $b = 12$, $n = 8$, $p_1 = p_2 = p_3 = p_4 = 13$, and $p_5 = p_6 = p_7 = p_8 = 5$. It is easy to see that the minimum number of bins for this instance is 4; see, for example, that job J_i and job J_{i+4} are packed into bin B_i for $i = 1, \ldots, 4$. The corresponding makespan is 729. However, if we assign job J_i to the ith bin for $i = 1, \ldots, 4$ and assign the remaining jobs to the fifth bin, we get an optimal schedule with a makespan of 728. Clearly, this implies that it is not necessary for an optimal schedule of the considered scheduling problem to have the minimum number of bins.

6. Conclusion

In this paper, we revisited the single machine scheduling problem considered in Xu et al. [14]; we provided an approximation named FFD-LS2I with a worst-case bound of 2 for the case $b > 1$. We showed that the FFD-LS2I algorithm is the best possible algorithm for the case $b > 1$ and that the FFD-LS algorithm is the best possible algorithm for the case $b \leq 1$ both from the worst-case bound point of view. We also give an answer to another open problem proposed in Xu et al. [14].

Future research may focus on the design of approximation algorithms with a lower time complexity while maintaining the worst-case bound of 2. The other maintenance duration functions such as convex function and concave function are also worth considering.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported in part by the National Natural Science Foundation of China (71201022) and the Natural Science Foundation of Jiangxi Province (20122BAB201010).

References

[1] G. Schmidt, "Scheduling with limited machine availability," *European Journal of Operational Research*, vol. 121, no. 1, pp. 1–15, 2000.

[2] C.-Y. Lee, "Machine scheduling with availability constraints," in *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, J. Y.-T. Leung, Ed., pp. 22.1–22.13, CRC Press, Boca Raton, Fla, USA, 2004.

[3] M. Ji, Y. He, and T. C. E. Cheng, "Single-machine scheduling with periodic maintenance to minimize makespan," *Computers & Operations Research*, vol. 34, no. 6, pp. 1764–1770, 2007.

[4] Y. Ma, C. Chu, and C. Zuo, "A survey of scheduling with deterministic machine availability constraints," *Computers & Industrial Engineering*, vol. 58, no. 2, pp. 199–211, 2010.

[5] K. Sun and H. Li, "Scheduling problems with multiple maintenance activities and non-preemptive jobs on two identical parallel machines," *International Journal of Production Economics*, vol. 124, no. 1, pp. 151–158, 2010.

[6] C.-J. Hsu, T. C. E. Cheng, and D.-L. Yang, "Unrelated parallel-machine scheduling with rate-modifying activities to minimize the total completion time," *Information Sciences*, vol. 181, no. 20, pp. 4799–4803, 2011.

[7] S. Bock, D. Briskorn, and A. Horbach, "Scheduling flexible maintenance activities subject to job-dependent machine deterioration," *Journal of Scheduling*, vol. 15, pp. 565–578, 2012.

[8] V. Gordon, V. Strusevich, and A. Dolgui, "Scheduling with due date assignment under special conditions on job processing," *Journal of Scheduling*, vol. 15, pp. 447–456, 2012.

[9] B. Mor and G. Mosheiov, "Scheduling a maintenance activity and due-window assignment based on common flow allowance," *International Journal of Production Economics*, vol. 135, no. 1, pp. 222–230, 2012.

[10] D. Xu, M. Liu, Y. Yin, and J. Hao, "Scheduling tool changes and special jobs on a single machine to minimize makespan," *Omega*, vol. 41, pp. 299–304, 2013.

[11] D. Xu and D.-L. Yang, "Makespan minimization for two parallel machines scheduling with a periodic availability constraint: mathematical programming model, average-case analysis, and anomalies," *Applied Mathematical Modelling*, vol. 37, pp. 7561–7567, 2013.

[12] D.-L. Yang and S.-J. Yang, "Unrelated parallel-machine scheduling with multiple rate-modifying activities," *Information Sciences*, vol. 235, pp. 280–286, 2013.

[13] S.-J. Yang, "Unrelated parallel-machine scheduling with deterioration effects and deteriorating multimaintenance activities for minimizing the total completion time," *Applied Mathematical Modelling*, vol. 37, pp. 2995–3005, 2013.

[14] D. Xu, Y. Yin, and H. Li, "Scheduling jobs under increasing linear machine maintenance time," *Journal of Scheduling*, vol. 13, no. 4, pp. 443–449, 2010.

[15] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.

[16] E. G. Coffman Jr., M. R. Garey, and D. S. Johnson, "Approximation algorithms for bin packing: a survey," in *Approximation Algorithms for NP-Hard Problems*, D. S. Hochbaum, Ed., pp. 46–93, PWS, Boston, Mass, USA, 1997.

[17] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Springer, New York, NY, USA, 4th edition, 2012.

[18] D. Simchi-Levi, "New worst-case results for the bin-packing problem," *Naval Research Logistics*, vol. 41, no. 4, pp. 579–585, 1994.

[19] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, NY, USA, 1979.