ANZIAM J. 58(2017), 314–323
doi:10.1017/S1446181117000141

SCHEDULING ON TWO PARALLEL MACHINES WITH TWO DEDICATED SERVERS

YIWEI JIANG[1,2], PING ZHOU[3], HUIJUAN WANG[2] and JUELIANG HU ✉[2]

(Received 24 May, 2016; accepted 15 October, 2016; first published online 25 May 2017)

Abstract

We study a nonpreemptive scheduling on two parallel identical machines with a dedicated loading server and a dedicated unloading server. Each job has to be loaded by the loading server before being processed on one of the machines and unloaded immediately by the unloading server after its processing. The loading and unloading times are both equal to one unit of time. The goal is to minimize the makespan. Since the problem is NP-hard, we apply the classical list scheduling and largest processing time heuristics, and show that they have worst-case ratios, $8/5$ and $6/5$, respectively.

2010 *Mathematics subject classification*: primary 90B35; secondary 90C27.

Keywords and phrases: scheduling, server, algorithm, worst-case ratio, makespan.

1. Introduction

In this paper, we consider a parallel-machine scheduling problem with two dedicated servers: a loading server and an unloading server, which are used to load and unload jobs onto and from the machines before and after their processing, respectively. The applications of the scheduling with servers can be found in different areas, such as flexible manufacturing systems (FMSs) [9], cellular manufacturing [1], the semiconductor industry [8] and the steel-making industry [15].

The problem can be described as follows. We are given a sequence $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ of n independent jobs, which must be processed on one of the two identical machines M_1 and M_2. Job J_j is associated with a loading time s_j, a processing time p_j and an unloading time t_j, all of which are positive integers. Each job J_j has to be loaded by the loading server S_1 before its processing and the loading time is s_j. Similarly, after finishing processing, a job has to be unloaded by the unloading server S_2 from the machine and the unloading time is t_j. Once the loading (unloading)

[1]College of Finance and Trade, Ningbo Dahongying University, Ningbo 315175, China;
e-mail: ywjiang@zstu.edu.cn.
[2]School of Sciences, Zhejiang Sci-Tech University, Hangzhou 310018, China; e-mail: hujlhz@163.com.
[3]College of Humanities, Zhejiang Business College, Hangzhou 310053, China.

is finished, the loading (unloading) server is free to load (unload) another job. The servers cannot load or unload jobs onto or from a machine when the machine is busy processing a job. Preemption is not considered in our paper. The objective is to minimize the makespan, that is, the completion time of the last job. Using the three-field notation for describing scheduling problems, we denote our problem as $P2, S2|s_j, t_j|C_{\max}$ for $j = 1, 2, \ldots, n$.

We use the *worst-case ratio* to measure the performance of an off-line approximation algorithm A. The worst-case ratio can be defined as the smallest number ρ such that for any $\mathcal{J}$, $C^A(\mathcal{J}) \leq \rho C^*(\mathcal{J})$, where $C^A(\mathcal{J})$ (or C^A in short) denotes the makespan produced by A and $C^*(\mathcal{J})$ (or C^* in short) denotes the optimal makespan. A worst-case ratio ρ is called *tight* if there is an instance $\mathcal{J}$ such that $C^A(\mathcal{J}) = \rho C^*(\mathcal{J})$ or $C^A(\mathcal{J})/C^*(\mathcal{J}) \to \rho$ as some parameters of the instance tend to a finite value or infinity.

Research on parallel-machine scheduling with servers is abundant. Kravchenko and Werner [10] showed that the problem $P2, S1|s_j = 1|C_{\max}$ is binary NP-hard and proposed a pseudo-polynomial algorithm. Hall et al. [3] showed that the problem $P2, S1|p_j = 1|C_{\max}$ can be solved in polynomial time and the problem $P2, S1|s_j = s|C_{\max}$ is strongly NP-hard. Brucker et al. [2] showed that the problem $P2, S1|p_j = p|C_{\max}$ is NP-hard. Kim and Lee [8] gave several heuristics for the problem with an arbitrary number of machines. Jiang et al. [4] considered the preemptive variant of the scheduling, that is, $P2, S1|pmpt|C_{\max}$. For an on-line variant of the problem, Zhang and Wirth [16] applied the *LS (list scheduling)* heuristic for three special cases where jobs arrive over time. Su [12] applied the *LPT (largest processing time)* heuristic for the on-line problem where jobs arrive over list. For the variant $P2, S_1|s_j = 1|C_{\max}$, Jiang et al. [6] showed that the competitive ratio of LS is at least $8/5$ and provided an on-line algorithm with competitive ratio of $11/7$. In addition, Ou et al. [11] and Werner and Kravchenko [14] studied the problem with multiple servers. Beside the objective of minimizing the makespan, Hall et al. [3] considered the objectives of minimizing the maximum lateness and the total completion time. Wang and Cheng [13] proposed an approximation algorithm for the problem to minimize the total weighted completion time.

For the problem involving both loading and unloading operations, Jiang et al. [7] considered the case $P2, S1|s_j = t_j = 1|C_{\max}$ and showed that LS and LPT have tight worst-case ratios of $12/7$ and $4/3$, respectively. For the on-line variant of $P2, S_1|s_j = t_j = 1|C_{\max}$, Jiang et al. [6] presented an on-line algorithm with competitive ratio of $5/3$, which is an improvement of LS. For the preemptive variant of the problem, Jiang et al. [5] provided an $O(n \log n)$ solution algorithm. Besides, Xie et al. [15] considered a relaxation of the problem in which the unloading operation can be delayed after a job finishes processing.

In this paper, we consider parallel-machine scheduling with two servers. Here the server cannot only load the jobs, but also unload the jobs similar to those in all the above-mentioned studies. Our problem can be reduced to the classical problem $P2|C_{\max}|$. So, we assume that the servers are different, namely, one is dedicated to load a job onto the machine before its processing (loading server) and the other is dedicated to unload the job after its processing (unloading server). The loading and unloading times are both equal to one time unit. Preemption is not considered here. Since our problem, denoted by $P2, S2|s_j = t_j = 1|C_{\max}$, is NP-hard, we consider approximation algorithms and show that the worst-case ratios of the heuristics LS and LPT are at most $8/5$ and $6/5$, respectively.

The rest of the paper is organized as follows. In Section 2 we provide some preliminaries and in Section 3 we analyse the structure of LS and derive its worst-case ratio. In Section 4 we consider the performance of LPT. Finally, we conclude the paper in Section 5.

2. Preliminaries

In this section, we present the lower bounds of the optimal makespan for our problem and introduce a couple of notations and definitions.

Let e_j denote the execution time of J_j, that is, $e_j = s_j + p_j + t_j$, where the loading time $s_j = 1$, the unloading time $t_j = 1$ and the processing time p_j is a positive integer. So, we have $e_j = p_j + 2 \geq 3$. Let $E = \sum_{i=1}^{n} e_i$ be the total execution time of all the jobs and $e_{\max}$ be the largest execution time among all jobs, that is, $e_{\max} = \max_{1\leq i\leq n}\{e_i\}$. Denote by C^A and C^* the makespan produced by algorithm A and the optimal makespan, respectively. In the subsequent algorithms, let l_j^1 and l_j^2 be the current completion times of M_1 and M_2, respectively, before scheduling job J_j. We obtain the following result, similar to the result on the optimal makespan provided by Jiang et al. [7].

LEMMA 2.1. *For the problem $P2, S2|s_j = t_j = 1|C_{\max}$, we have $C^* \geq \max\{e_{\max}, (E + 2)/2\}$.*

PROOF. It is clear that $C^* \geq e_{\max}$. On the other hand, when the loading server loads the first job onto one of the machines, the other machine must be idle. Similarly, a machine must be idle when the unloading server unloads the last completed job. Thus, there are at least two idle time units in the optimal scheduling. Together with the total execution time E, we can conclude that the optimal makespan is at least $(E + 2)/2$ and the result holds. □

We introduce two definitions to simplify the expression in the sequel, which are illustrated in Figure 1.

DEFINITION 2.2. (i) Two adjacent time units are called a double unloading time unit if each is a job unloading time. The end time of a double unloading time unit is called an EDT point. (ii) The time zero is defined as the first EDT point.

DEFINITION 2.3. (i) A block $\mathcal{B}$ in a schedule is defined as a period of time between two adjacent EDT points. (ii) The period of time from the last EDT to the end of the schedule is called an unfinished block $\mathcal{B}'$.

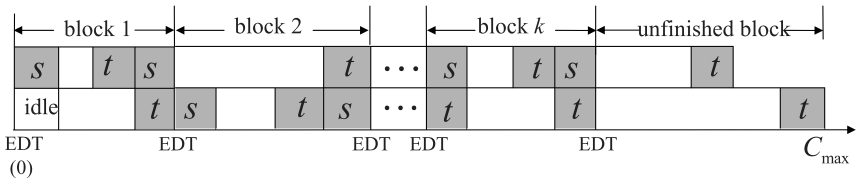

FIGURE 1. Definitions of EDT, block and unfinished block; structure of LS schedule.

3. List scheduling heuristic

In this section, we apply the LS heuristic to tackle our problem. The LS assigns the current job to the first available machine at the earliest possible time, which can be described in detail as follows.

Algorithm 1 List scheduling

```
STEP 1 Initially set $j = 1$ and $l_j^1 = l_j^2 = 0$.
STEP 2 (Job assignment) If $l_j^1 \leq l_j^2$, assign $J_j$ to machine $M_1$; otherwise, assign it to $M_2$.
STEP 3 (Job execution) The loading of the job $J_j$ starts at the earliest time such that its
       unloading time cannot be the same as that of the job processing on the other
       machine.
STEP 4 If no new job arrives, stop. Otherwise, $j = j + 1$, return to STEP 2.
```

PROPOSITION 3.1. *The structure of the schedule σ by LS must be one of the following three types:*

(i) *an unfinished block, that is, $\sigma = \mathcal{B}'$;*
(ii) *a number of successive blocks, that is, $\sigma = (\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k)$;*
(iii) *a number of successive blocks followed by an unfinished block, that is, $\sigma = (\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k, \mathcal{B}')$.*

PROOF. The proof follows from the LS rule and Definitions 2.2 and 2.3. Generally, we obtain the structure of schedule σ as shown in Figure 1. □

We provide an important property of the block $\mathcal{B}$ in the schedule produced by LS.

PROPOSITION 3.2. *For any block $\mathcal{B}$, there is at most one idle time unit in the middle of $\mathcal{B}$.*

PROOF. By the LS rule, we always schedule the job as early as possible. Therefore, we conclude that no new idle time is introduced before the formation of the block $\mathcal{B}$ (that is, it is currently an unfinished block). We consider the assignment of any job $J_k \in \mathcal{B}$, except the first job in the block $\mathcal{B}$. Clearly, if $e_k - 1 \leq |l_k^1 - l_k^2| \leq e_k + 1$, job J_k must be scheduled at the time $\min\{l_k^1, l_k^2\}$, without introducing new idle time. On the other hand, if $e_k = |l_k^1 - l_k^2|$, for job J_k, LS must schedule it at the time $\min\{l_k^1, l_k^2\} + 1$

and an idle time unit is consequently introduced. At the same time, however, a block is formed after scheduling this job. Hence, there is at most one idle time unit in the middle of the block and the proof is complete. □

Let $r(\mathcal{B})$ and $r(\mathcal{B}')$ be the numbers of idle time units in the block $\mathcal{B}$ and the unfinished block $\mathcal{B}'$, respectively.

COROLLARY 3.3. *For the general LS schedule $\sigma = (\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k, \mathcal{B}')$, we have:*

(i) *$r(\mathcal{B}_1) = 1$ or 2; furthermore, $r(\mathcal{B}_1) = 2$ or 3 if $r(\mathcal{B}') = \emptyset$ and $k = 1$;*
(ii) *$r(\mathcal{B}_i) = 0$ or 1 for any $2 \leq i \leq k$; furthermore, $r(\mathcal{B}_k) = 1$ or 2 if $r(\mathcal{B}') = \emptyset$;*
(iii) *$r(\mathcal{B}') = 0$ if $k > 0$ and $r(\mathcal{B}') = 1$ if $k = 0$.*

PROOF.

(i) It is clear that there is an idle time unit at the beginning of the block. Then we have $r(\mathcal{B}_1) = 1$ or 2 due to Proposition 3.2. Moreover, if $r(\mathcal{B}') = \emptyset$ and $k = 1$, that is, the obtained LS schedule consists of a single block, then we have $r(\mathcal{B}_1) = 2$ or 3 with the fact that there is an idle time unit at the end of the block.
(ii) For any $2 \leq i \leq k$, we obtain that the idle time produced at the end of block $\mathcal{B}_i$ can be used to load the first job in the next block $\mathcal{B}_{i+1}$ as shown in Figure 1. This implies that there is no idle time at the beginning and the end of the block. Thus, we have $r(\mathcal{B}_i) = 0$ or 1 by Proposition 3.2. If $r(\mathcal{B}') = \emptyset$, there is an idle time unit at the end of the block $\mathcal{B}_k$ and thus $r(\mathcal{B}_k) = 1$ or 2.
(iii) Note that there is no more idle time before the formation of a block. If $k > 0$, there is no idle time at the beginning of $\mathcal{B}'$ and thus $r(\mathcal{B}') = 0$. If $k = 0$, that is, $\sigma = \mathcal{B}'$, there thus is an idle time unit at the beginning of $\mathcal{B}'$, that is, $r(\mathcal{B}') = 1$. □

We now show that the worst-case ratio of LS is $8/5$. Let $l(\mathcal{B})$ and $e(\mathcal{B})$ be the time length of a block $\mathcal{B}$ and the total execution time of all the jobs in $\mathcal{B}$, respectively.

THEOREM 3.4. *The worst-case ratio of LS is at most* $8/5$.

PROOF. Let J_l be the last completed job and T be the start time of J_l; then $C^{\mathrm{LS}} = T + e_l$. We consider two cases according to the structure of the schedule σ by Proposition 3.1.

Case 1. $\sigma = \mathcal{B}'$. This is a single unfinished block. From Corollary 3.3(iii), we have $r(\mathcal{B}') = 1$. From Lemma 2.1, it follows that the total execution time of all the jobs is at least $2T - 1 + e_l$ and $C^* \geq \max\{(2T + e_l + 1)/2, e_l\} > \max\{(2T + e_l)/2, e_l\}$. Hence,

$$\frac{C^{\mathrm{LS}}}{C^*} \leq \frac{T + e_l}{\max\{(2T + e_l)/2, e_l\}} \leq \begin{cases} \dfrac{2T + 2e_l}{2T + e_l} \leq \dfrac{3}{2} < \dfrac{8}{5}, & e_l \leq 2T, \\ \dfrac{T + e_l}{e_l} \leq \dfrac{3}{2} < \dfrac{8}{5}, & e_l > 2T. \end{cases}$$

Case 2. $\sigma = (\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k, \mathcal{B}')$. Let $X = \sum_{i=1}^{k} l(\mathcal{B}_i)$ be the total length of all the k blocks. Note that $l(\mathcal{B}_1) \geq 4$ and $l(\mathcal{B}_i) \geq 3$ for any $2 \leq i \leq k$, since $e_i \geq 3$ for all $1 \leq i \leq n$.

Thus,

$$X \geq 3k + 1. \tag{3.1}$$

If $\mathcal{B}' = \emptyset$, the total idle time is $\sum_{i=1}^{k} r(\mathcal{B}_i) \leq k + 2$ from Corollary 3.3(i) and (ii), implying that $E \geq 2X - (k + 2)$. Noting that $C^{\mathrm{LS}} = X$ and $C^* \geq \{2X - (k + 2) + 2\}/2$ due to Lemma 2.1,

$$\frac{C^{\mathrm{LS}}}{C^*} \leq \frac{2X}{2X - k} \leq \frac{2X}{5X/3 + 1/3} < \frac{6}{5} < \frac{8}{5}.$$

If $\mathcal{B}' \neq \emptyset$, then, from Corollary 3.3, the total idle time is $\sum_{i=1}^{k} r(\mathcal{B}_i) \leq k + 1$. Note that $\mathcal{B}'$ is scheduled after all the blocks, so the last completed job J_l must belong to $\mathcal{B}'$.

If J_l is one of the first two jobs in $\mathcal{B}'$, we have $C^{\mathrm{LS}} = T + e_l \leq X + e_l$ and $E \geq 2X - (k + 1) + e_l \geq 5X/3 - 2/3 + e_l$, due to the inequality (3.1). From Lemma 2.1,

$$C^* \geq \max\left\{\frac{5X/3 - 2/3 + e_l + 2}{2}, e_l\right\} \geq \max\left\{\frac{5X/3 + 4/3 + e_l}{2}, e_l\right\}$$

and, thus,

$$\frac{C^{\mathrm{LS}}}{C^*} \leq \frac{X + e_l}{\max\{(5X/3 + 4/3 + e_l)/2, e_l\}} < \frac{8}{5}.$$

On the other hand, if J_l is not one of the first two jobs in $\mathcal{B}'$, then there is at least one job J_x in $\mathcal{B}'$ that is scheduled before J_l, that is, $\Delta = T - X \geq e_x - 1 \geq 2$. Let $e(\Delta)$ be the total execution time of the jobs processed on the two machines during the period of time from X to T. Since there is no idle time in the middle of $\mathcal{B}'$, we have $e(\Delta) = 2\Delta$ and thus $E \geq 2X - (k + 1) + e(\Delta) + e_l \geq 5X/3 - 2/3 + e_l + 2\Delta > 5(X + \Delta)/3 - 2/3 + e_l$ due to (3.1). Noting that $C^{\mathrm{LS}} = T + e_l = X + \Delta + e_l$ and

$$C^* \geq \max\left\{\frac{5(X + \Delta)/3 + e_l + 4/3}{2}, e_l\right\},$$

$$\frac{C^{\mathrm{LS}}}{C^*} \leq \frac{(X + \Delta) + e_l}{\max\{(5(X + \Delta)/3 + e_l + 4/3)/2, e_l\}} < \frac{8}{5}.$$

The proof is now complete. □

4. LPT heuristic

In this section, we apply LPT to tackle our problem and show that its worst-case ratio is at most 6/5. We describe LPT in detail in Algorithm 2.

Algorithm 2 Largest processing time

```
1. Sort all the jobs such that $e_1 \geq e_2 \geq \cdots \geq e_n$.
2. For any job $J_i$, $1 \leq i \leq n$, schedule it by LS.
```

Note that the schedule σ produced by LPT is a list schedule with the jobs originally listed in nonincreasing order of their processing times, so Propositions 3.1 and 3.2, and Corollary 3.3, still apply in this section.

LEMMA 4.1. *If $\sigma = \mathcal{B}'$, we have $C^{\mathrm{LPT}}/C^* \le 6/5$.*

PROOF. Let l_1 and l_2 be the completion times of the machines M_1 and M_2, respectively. Without loss of generality, we assume that $l_1 \le l_2$ and the last completion job is J_n. Let T be the start time of job J_n, that is, $C^{\mathrm{LPT}} = l_2 = T + e_n$. If there is only one job on the machine M_1, it is not difficult to obtain that the schedule generated by LPT is optimal. So, we assume that at least two jobs are processed on the machine M_1 and thus $l_1 \ge 2e_n$. Since $\sigma = \mathcal{B}'$, we conclude that there is only one idle time unit from Corollary 3.3(iii), which implies that $E = l_1 + l_2 - 1$ and thus

$$C^* \ge \frac{l_1 + l_2 - 1 + 2}{2} = \frac{l_1 + T + e_n + 1}{2}.$$

Noting that no block is formed, the difference between the completion times of two machines is at least two time units after scheduling any job. Then we have $T \le l_1 - 2$. Hence,

$$\frac{C^{\mathrm{LPT}}}{C^*} \le \frac{2T + 2e_n}{l_1 + T + e_n + 1} \le \frac{2(l_1 - 2) + 2e_n}{l_1 + l_1 - 2 + e_n + 1} = \frac{2l_1 + 2e_n - 4}{2l_1 + e_n - 1} \le \frac{6e_n - 4}{5e_n - 1} < \frac{6}{5}. \qquad \square$$

Next, we consider the case where the schedule σ consists of k $(k \ge 1)$ blocks $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k$. Similar to the previous section, let $l(\mathcal{B}_i)$ and $e(\mathcal{B}_i)$ be the time length of a block $\mathcal{B}_i$ and the total execution time of all the jobs in $\mathcal{B}_i$, respectively, and denote $X = \sum_{i=1}^{k} l(\mathcal{B}_i)$. Let q be the execution time of the smallest job in all the k blocks. There are at least k jobs on each machine, because we have k blocks. Since there is at least one idle time at the beginning of the schedule, we conclude that

$$X \ge kq + 1. \tag{4.1}$$

We first consider the case where $\mathcal{B}' = \emptyset$.

LEMMA 4.2. *If $\mathcal{B}' = \emptyset$, that is, $\sigma = (\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k)$, then $C^{\mathrm{LPT}}/C^* \le 6/5$.*

PROOF. Suppose that there is only one block, that is, $k = 1$. By Corollary 3.3(i), there are at most three idle time units. It implies that $C^{\mathrm{LPT}} \le (E + 3)/2$ and, thus,

$$\frac{C^{\mathrm{LPT}}}{C^*} \le \frac{(E + 3)/2}{(E + 2)/2} \le 6/5,$$

since $E \ge 3$. So, we assume that $k \ge 2$ below.

From Corollary 3.3(i) and (ii), we obtain that $r(\mathcal{B}_1) \le 2$, $r(\mathcal{B}_k) \le 2$ and $r(\mathcal{B}_i) \le 1$ for any $2 \le i \le k - 1$. It means that the total idle time is at most $k + 2$ time units, that is, $E \ge 2X - (k + 2)$. From the inequality (4.1), $C^{\mathrm{LPT}} = X \ge kq + 1 > kq$. Thus,

$$\frac{C^{\mathrm{LPT}}}{C^*} \le \frac{X}{(E + 2)/2} \le \frac{X}{(2X - k)/2} = \frac{1}{1 - (k/2X)} \le \frac{1}{1 - (1/2q)} \le \frac{6}{5},$$

where the last inequality holds because $q \ge 3$. $\square$

We next consider the case where $\mathcal{B}' \ne \emptyset$, which implies that the last completed job J_l belongs to $\mathcal{B}'$. By LPT, the execution time of J_l is not greater than any one of the jobs in $\bigcup_{1\le i\le k}\mathcal{B}_i$, that is,

$$e_l \le q. \tag{4.2}$$

Before giving the result for this case, we draw a conclusion for the job J_l as follows.

Lemma 4.3. *If $\mathcal{B}' \neq \emptyset$, the job J_l cannot be the second job to be processed in $\mathcal{B}'$.*

Proof. Denote by J_x and J_y the first job and the second job, respectively, to be processed in $\mathcal{B}'$. Note that the job J_x must be loaded at time $X-1$, because there is an idle time at the end of the block $\mathcal{B}_k$ on one machine, while the job J_y must be scheduled on the other machine, which is available at time X. Note that $e_x \ge e_y$ by the LPT rule. It is clear that the job J_y is completed earlier than the job J_x if $e_y \le e_x - 3$. On the other hand, if $e_x - 2 \le e_y \le e_x$, J_x and J_y form a new block, which contradicts the fact that both jobs are in $\mathcal{B}'$. Hence, the second job J_y must be completed earlier than the first job J_x, that is, J_l cannot be the second job in $\mathcal{B}'$. □

Lemma 4.4. *If $\sigma = (\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_k, \mathcal{B}')$, then $C^{\mathrm{LPT}}/C^* \le 6/5$.*

Proof. We begin with a special case where J_l is the first job in $\mathcal{B}'$. Then

$$C^{\mathrm{LPT}} = X + e_l - 1.$$

Two cases are considered according to the number of the blocks.

Case (a). Let $k = 1$. If the block $\mathcal{B}_1$ exactly consists of two jobs, that is, J_1 and J_2, it implies that J_l is the third job J_3. We focus on the assignment of the job J_3. If J_3 and J_2 are scheduled on the same machine, we must have $e_1 = e_2 + 2$ and $C^{\mathrm{LPT}} = e_2 + e_3 + 1$. On the other hand, if J_3 and J_1 are scheduled on the same machine, then we have $e_1 - 1 \le e_2 \le e_1$ and thus $C^{\mathrm{LPT}} = e_1 + e_3 \le e_2 + e_3 + 1$. Obviously, $C^* \ge e_2 + e_3$. Hence, $C^{\mathrm{LPT}}/C^* \le (e_2 + e_3 + 1)/(e_2 + e_3) < 6/5$, since $e_2 \ge e_3 \ge 3$.

If there are at least three jobs in the block, then $X = l(\mathcal{B}_1) \ge 2q \ge 2e_l$. By Corollary 3.3(i), there are at most three idle time units in the block before scheduling J_l, so it follows that $E \ge 2X - 3 + e_l$ and $C^* > (2X + e_l - 1)/2$. Therefore,

$$\frac{C^{\mathrm{LPT}}}{C^*} \le \frac{X + e_l - 1}{(2X + e_l - 1)/2} \le \frac{2X + 2e_l - 2}{2X + e_l - 1} \le \frac{6e_l - 2}{5e_l - 1} < \frac{6}{5}.$$

Case (b). Let $k \ge 2$. From the inequalities (4.1) and (4.2), we have $X \ge kq + 1 \ge ke_l + 1$. By Corollary 3.3(i) and (ii), there are at most $k + 2$ idle time units in all k blocks before scheduling J_l. Thus, we have $E \ge 2X - (k + 2) + e_l$ and $C^* > (2X + e_l - k)/2$, from which it follows that

$$\begin{aligned}\frac{C^{\mathrm{LPT}}}{C^*} \le \frac{X + e_l - 1}{(2X + e_l - k)/2} &\le \frac{2X + 2e_l - 2}{2X + e_l - k}\\ &\le \frac{2ke_l + 2 + 2e_l - 2}{2ke_l + 2 + e_l - k}\\ &= \frac{2e_l k + 2e_l}{(2e_l - 1)k + e_l + 2} \le \frac{6}{5},\end{aligned}$$

where the last inequality holds because $k \ge 2$ and $e_l \ge 3$.

Now we consider the case where J_l is not the first job in $\mathscr{B}'$. Together with Lemma 4.3, we assume that J_l is not the first two jobs in $\mathscr{B}'$. Let T be the start time of the job J_l and $\Delta = T - X$, that is, $C^{\text{LPT}} = T + e_l = X + \Delta + e_l$.

Moreover, at least one job in $\mathscr{B}'$ is processed during the time interval $[X, T]$ on the machine processing J_l. Then we have $\Delta = T - X \geq e_l - 1$. Noting that in $\mathscr{B}'$, the difference between the completion times of two machines is at least two time units after scheduling any job, the completion time of the machine without processing J_l is at least $T + 2$. Besides, it is easy to obtain that there are at most $k + 1$ idle time units by Corollary 3.3. Therefore, we have $E \geq (T + 2) + (T + e_l) - (k + 1) = 2T + e_l - k + 1$ and thus $C^* \geq (2T + e_l - k + 3)/2 = (2X + 2\Delta + e_l - k + 3)/2$. Hence, by the inequalities (4.1) and (4.2),

$$\begin{aligned}\frac{C^{\text{LPT}}}{C^*} &\leq \frac{2X + 2\Delta + 2e_l}{2X + 2\Delta + e_l - k + 3}\\ &\leq \frac{2ke_l + 2 + 2(e_l - 1) + 2e_l}{2ke_l + 2 + 2(e_l - 1) + e_l - k + 3}\\ &= \frac{2ke_l + 4e_l}{2ke_l + 3e_l - k + 3} < \frac{6}{5},\end{aligned}$$

where the last inequality holds because $k \geq 1$ and $e_l \geq 3$. □

Summarizing the results from Lemmas 4.1, 4.2 and 4.4, we conclude the section with the following result.

THEOREM 4.5. *The worst-case ratio of LPT is at most* $6/5$.

5. Conclusion

We studied a parallel-machine scheduling with two servers to minimize the makespan, where each job has to be loaded by a loading server and unloaded by an unloading server, respectively, before and after being processed on one of the two machines. We assumed that both loading and unloading take one unit of time, and showed that LS and LPT have worst-case ratios of $8/5$ and $6/5$, respectively. Note that the bounds of our two algorithms are not tight, so it is very interesting to find their tight bounds. In addition, it is also worth studying the performance of LS and LPT for the scheduling on m machines.

Acknowledgements

The work is partially supported by China NSF, Grant Nos. 11571013 and 11471286, and Zhejiang Provincial NSF of China under Grant No. LY14A010031.

References

[1] G. Batur, O. Karasan and M. Akturk, “Multiple part-type scheduling in flexible robotic cells”, *Int. J. Product. Econ.* **135** (2012) 726–740; doi:10.1016/j.ijpe.2011.10.006.

[2] P. Brucker, C. Dhaenens-Flipo, S. Knust, S. A. Kravchenko and F. Werner, “Complexity results for parallel machine problems with a single server”, *J. Sched.* **5** (2002) 429–457; doi:10.1002/jos.120.

[3] N. Hall, C. Potts and C. Sriskandarajah, “Parallel machine scheduling with a common server”, *Discrete Appl. Math.* **102** (2000) 223–243; doi:10.1016/S0166-218X(99)00206-1.

[4] Y. Jiang, J. Dong and M. Ji, “Preemptive scheduling on two parallel machines with a single server”, *Comput. Ind. Eng.* **66** (2013) 514–518; doi:10.1016/j.cie.2013.07.020.

[5] Y. Jiang, H. Wang and P. Zhou, “An optimal preemptive algorithm for the single-server parallel-machine scheduling with loading and unloading times”, *Asia-Pac. J. Oper. Res.* **32** (2014) 11 pages; doi:10.1142/S0217595914500390.

[6] Y. Jiang, F. Yu, P. Zhou and J. Hu, “Online algorithms for scheduling on two parallel machines with a single server”, *Int. Trans. Oper. Res.* **22** (2015) 913–927; doi:10.1111/itor.12136.

[7] Y. Jiang, Q. Zhang, J. Hu, J. Dong and M. Ji, “Single-server parallel-machine schduling with loading and unloading times”, *J. Comb. Optim.* **30** (2015) 201–213; doi:10.1007/s10878-014-9727-z.

[8] M. Y. Kim and Y. H. Lee, “MIP models and hybrid algorithm for minimizing the makespan of parallel machines scheduling problem with a single server”, *Comput. Oper. Res.* **39** (2012) 2457–2468; doi:10.1016/j.cor.2011.12.011.

[9] C. Koulamas, “Scheduling two parallel semiautomatic machines to minimize machine interference”, *Comput. Oper. Res.* **23** (1996) 945–956; doi:10.1016/0305-0548(96)00011-1.

[10] S. Kravchenko and F. Werner, “Parallel machine scheduling problems with a single server”, *Math. Comput. Modelling* **26** (1997) 1–11; doi:10.1016/S0895-7177(97)00236-7.

[11] J. Ou, X. Qi and C. Lee, “Parallel machine scheduling with multiple unloading servers”, *J. Sched.* **13** (2010) 213–226; doi:10.1007/s10951-009-0104-1.

[12] C. Su, “Online LPT algorithms for parallel machines scheduling with a single server”, *J. Comb. Optim.* **26** (2013) 480–488; doi:10.1007/s10878-011-9441-z.

[13] G. Wang and T. C. E. Cheng, “An approximation algorithm for parallel machine scheduling with a common server”, *J. Oper. Res. Soc.* **52** (2001) 234–237; doi:10.1057/palgrave.jors.2601074.

[14] F. Werner and S. Kravchenko, “Scheduling with multiple servers”, *Autom. Remote Control* **71** (2010) 2109–2121; doi:10.1134/S0005117910100103.

[15] X. Xie, Y. Li, H. Zhou and Y. Zheng, “Scheduling parallel machines with a single server”, in: *Measurement, information and control (MIC)* (IEEE, Harbin, China, 2012) 453–456; doi:10.1109/MIC.2012.6273340.

[16] L. Zhang and A. Wirth, “On-line scheduling of two parallel machines with a single server”, *Comput. Oper. Res.* **36** (2009) 1529–1553; doi:10.1016/j.cor.2008.02.015.