J Sched
DOI 10.1007/s10951-017-0546-9

An algorithm for multi-agent scheduling to minimize the makespan on m parallel machines

Manzhan Gu[1] · **Jinwei Gu**[2] · **Xiwen Lu**[3]

Abstract This paper considers a multi-agent scheduling problem, in which each agent has a set of non-preemptive jobs, and jobs of all agents are to be processed on m identical parallel machines. The objective is to find a schedule to minimize the makespan of each agent. For an agent, the definition of α point is introduced, based on which an approximation algorithm is proposed for the problem. In the obtained schedule, the agent with the ith smallest α point value is the ith completed agent, and the agent's completion time is at most $i+\left(\frac{1}{3}-\frac{1}{3m}\right)$ times its minimum makespan. Finally, we show the performance analysis is tight.

Keywords Scheduling · Multi-agent · LPT · Makespan

1 Introduction

This paper studies the multi-agent scheduling problem. There are g agents, and each agent has a set of jobs, which are to be scheduled on m identical parallel machines. Each agent wants to minimize its own makespan. Note it is impossible to find a schedule in which the makespan of each agent equals its *minimum makespan*, i.e., the optimal makespan for the problem with only this agent's jobs. Therefore, this paper aims to find a schedule to make each agent's makespan close to its minimum makespan as much as possible. Following the notation introduced by Agnetis et al. (2014), we denote by $Pm|CO|C^1_{\max}, C^2_{\max}, \ldots, C^g_{\max}$ the problem considered in this paper, where CO denotes the situation where each agent completes to finish its jobs as early as possible, and $C^i_{\max}(i = 1, 2, \ldots, g)$ is the makespan of agent i.

We employ the following definition introduced by Saule and Trystram (2009) to measure how each agent's makespan is close to its minimum makespan, which is also used to measure the performance of an approximation algorithm for the problem studied in this paper.

Definition 1 In the schedule σ^H generated by an algorithm H, suppose the makespan of the ith completed agent (i.e., the agent with the ith smallest makespan) satisfies

$$C_i^H \le \alpha_i \cdot C_i^*, \quad i = 1, 2, \ldots, g,$$

where C_i^H denotes the makespan of the ith completed agent in the schedule σ^H, C_i^* denotes the agent's minimum makespan and α_i is a constant. Then, we define H as a $(\alpha_1, \alpha_2, \ldots, \alpha_g)$-approximation algorithm and $(\alpha_1, \alpha_2, \ldots, \alpha_g)$ is the performance ratio vector of the algorithm H.

There is a large amount of research on multi-agent scheduling in the last decade. For the two-agent scheduling problem on a single machine, Agnetis et al. (2004) considered various models with some regular objective functions. Based on the Lagrangian method, Agnetis et al. (2009) presented branch-and-bound algorithms for three problems. Leung et al. (2010) studied the complexity of the preemptive problem with release dates. Li and Yuan (2012) and Fan et al. (2013) considered the batch scheduling problems.

Besides, some researchers considered the multi-agent scheduling problems on a single machine. Cheng et al. (2006) studied the problem with the objective to minimize the total

✉ Jinwei Gu
gujinwei1982@163.com

[1] Institute of Scientific Computation and Financial Data Analysis, School of Mathematics, Shanghai University of Finance and Economics, Shanghai 200433, China

[2] School of Economics and Management, Shanghai University of Electric Power, Shanghai 200090, China

[3] School of Science, East China University of Science and Technology, Shanghai 200237, China

Published online: 20 October 2017

weighted number of tardy jobs for each agent. Agnetis et al. (2007) provided complexity results for several multi-agent models. Besides, Cheng et al. (2008) analyzed the complexity of some special cases where each agent has max-form criterion.

The results on the approximation algorithms for multi-agent scheduling are limited. Saule and Trystram (2009) first studied approximation algorithms for multi-agent scheduling problems. For the problem $Pm|CO|C^1_{\max}, C^2_{\max}, \ldots, C^g_{\max}$, the authors proved there is no approximation algorithm with performance ratio vector better than $(1, 2, \ldots, g)$ even in the special case where $m = 1$ and $P = NP$. Furthermore, given an algorithm for the single-agent problem $P||C_{\max}$, assuming the performance ratio of the algorithm is ρ, the authors proposed a $(\rho, 2\rho, \ldots, g\rho)$-approximation algorithm for the multi-agent problem $Pm|CO|C^1_{\max}, C^2_{\max}, \ldots, C^g_{\max}$. Recently, regarding the two-machine problem $P2|CO|C^1_{\max}, C^2_{\max}, \ldots, C^g_{\max}$, Zhao et al. (2016) introduced a new approximation algorithm, and proved the algorithm has a better performance ratio vector $\left(1 + \frac{1}{6}, 2 + \frac{1}{6}, \ldots, g + \frac{1}{6}\right)$, which was further shown to be tight.

In addition, Lee et al. (2009) introduced a different definition of the performance ratio vector. In the paper Lee et al. (2009), the performance ratio vector of an algorithm is $(\beta_1, \beta_2, \ldots, \beta_g)$ means: in the generated schedule, the objective function value of agent i (not the ith completed agent) is at most β_i times its minimal objective function value. The authors considered the multi-agent problem with the objective to minimize the total weighted completion times and proposed a $(\beta_1, \beta_2, \ldots, \beta_g)$-approximation algorithm when $\sum_{i=1}^{g} \frac{1}{\beta_i} = 1$ and $\beta_i > 1$. The approximation ratio is shown to be the best possible.

This paper studies the m-machine problem $Pm|CO|C^1_{\max}, C^2_{\max}, \ldots, C^g_{\max}$. Based on the *longest processing time first* (LPT) rule, we propose an approximation algorithm for the problem and prove the performance ratio vector is

$$\left(1+\left(\frac{1}{3}-\frac{1}{3m}\right), 2+\left(\frac{1}{3}-\frac{1}{3m}\right), \ldots, g+\left(\frac{1}{3}-\frac{1}{3m}\right)\right),$$

where $\frac{1}{3} - \frac{1}{3m}$ is the relative error for LPT rule with respect to the single-agent problem $Pm||C_{max}$. We further give an instance to show the performance ratio vector is tight. The performance ratio vector is better than that in Saule and Trystram (2009). Compared to the algorithm given by Zhao et al. (2016) for the two-machine problem, we introduce a different algorithm, which is easier to be implemented. During the process of our algorithm, scheduler does not need to change the processing order of consecutive two agents, and he/she could determine the sequence of all agents with the α point values.

This paper is organized as follows. In Sect. 2, we formally introduce the problem and the notations used throughout this paper. In Sect. 3, we give the definition of the adjoint instance. An algorithm is introduced for the adjoint instance in Sect. 4. In Sect. 5, based on the results in Sect. 4, we propose a new approximation algorithm for the problem under study in this paper. Concluding remarks follow in Sect. 6.

2 Problem formulation and notation

There are g agents $\mathcal{G} = \{1, 2, \ldots, g\}$. Each agent $i (i \in \mathcal{G})$ has a set of jobs $\mathcal{J}^i = \{J^i_1, J^i_2, \ldots, J^i_{n_i}\}$, which are to be processed non-preemptively on m identical parallel machines. Denote by $p^i_j (i = 1, 2, \ldots, g, j = 1, 2, \ldots, n_i)$ the processing time of job J^i_j. The objective is to minimize the makespan of each agent, in which the makespan of an agent is the time when all its jobs have just been completed on the m machines. Denote by $\mathcal{I}_0$ the instance of problem $Pm|CO|C^1_{\max}, C^2_{\max}, \ldots, C^g_{\max}$. Given an instance $\mathcal{I}_0$, for each agent $i (i = 1, 2, \ldots, g)$, define I^i_0 as the instance of the single-agent problem $Pm||C_{\max}$ with only the agent's jobs.

Given a schedule σ for a multi-agent (or single-agent) scheduling problem on m machines, we define the following notations.

- $\boldsymbol{C^i_{\max}(\sigma)} (i = 1, 2, \ldots, g)$: The makespan of agent i in the schedule σ.
- $\boldsymbol{M_k(\sigma)} (k = 1, 2, \ldots, m)$: The machine with the kth largest processing load in schedule σ. If more than one machine have the same kth largest processing load, then choose a machine with the least number of jobs.
- $\boldsymbol{CM_k(\sigma)}$: The processing load on machine $M_k(\sigma)$ in schedule σ.
- $\boldsymbol{\Delta_{k,l}(\sigma)}$: The difference of the processing loads on the two machines $M_k(\sigma)$ and $M_l(\sigma)$ in schedule σ, i.e.,

$$\begin{aligned}\Delta_{k,l}(\sigma) &= |CM_k(\sigma) - CM_l(\sigma)| \\ &= CM_k(\sigma) - CM_l(\sigma), 1 \le k < l \le m.\end{aligned}$$

- $\boldsymbol{\Delta(\sigma)}$: The maximum difference of the processing loads on any two machines in schedule σ, i.e.,

$$\begin{aligned}\Delta(\sigma) &= \max_{1\le k<l\le m} \Delta_{k,l}(\sigma) \\ &= CM_1(\sigma) - CM_m(\sigma).\end{aligned}$$

Furthermore, regarding an instance I of the single-agent problem $Pm||C_{\max}$, we define the following notations.

- $\boldsymbol{C^*_{\max}(I)}$: The minimum makespan of instance I.
- $\boldsymbol{\sigma^{LPT}(I)}$: The schedule generated by LPT rule on instance I, i.e., the LPT schedule of instance I.
- $\boldsymbol{\mathcal{J}^{LPT}_k(I)}$: The set of jobs processed on the machine $M_k(\sigma^{LPT}(I))$ in schedule $\sigma^{LPT}(I)$ (possibly empty).

- $\boldsymbol{\mathcal{P}_k^{LPT}(I)}$: The total processing times of jobs in the job set $\mathcal{J}_k^{LPT}(I)$.
- $\boldsymbol{C_{\max}^{LPT}(I)}$: The makespan of the LPT schedule of instance I, which equals $\boldsymbol{\mathcal{P}_1^{LPT}(I)}$.
- $\boldsymbol{P_{last}^{LPT}(I)}$: The processing time of the last completed job on machine $M_1\left(\sigma^{LPT}(I)\right)$ in the schedule $\sigma^{LPT}(I)$. Obviously,

$$P_{last}^{LPT}(I) \geq \Delta\left(\sigma^{LPT}(I)\right). \tag{1}$$

Based on the notations defined above, we introduce the following two lemmas regarding the LPT schedule of the single-agent problem $Pm||C_{\max}$.

Lemma 1 (Pinedo 2011) *For the single-agent problem $Pm||C_{\max}$, if an optimal schedule is a schedule with at most two jobs on each machine, then the LPT schedule is optimal.*

Lemma 2 *For an instance I of the single-agent problem $Pm||C_{\max}$, assuming there are at least two jobs on the machine $M_1\left(\sigma^{LPT}(I)\right)$ in the LPT schedule $\sigma^{LPT}(I)$, we have*

(i) *if $C_{\max}^{LPT}(I) < 3P_{last}^{LPT}(I)$, then $C_{\max}^{*}(I) = C_{\max}^{LPT}(I)$;*
(ii) *if $C_{\max}^{LPT}(I) \geq 3P_{last}^{LPT}(I)$, then $C_{\max}^{*}(I) \geq 3P_{last}^{LPT}(I)$.*

Proof Based on instance I, we construct a new instance I' of the problem $Pm||C_{\max}$ by deleting all jobs with processing times less than $P_{last}^{LPT}(I)$ from I. Then

$$C_{\max}^{*}\left(I'\right) \leq C_{\max}^{*}(I). \tag{2}$$

According to the LPT rule, the new LPT schedule $\sigma^{LPT}\left(I'\right)$ could be obtained by removing jobs with processing times less than $P_{last}^{LPT}(I)$ from the schedule $\sigma^{LPT}(I)$. Hence,

$$P_{last}^{LPT}(I') = P_{last}^{LPT}(I), \tag{3}$$
$$C_{\max}^{LPT}\left(I'\right) = C_{\max}^{LPT}(I), \tag{4}$$

and $P_{last}^{LPT}(I')$ is the smallest processing time in the new instance I'.

We first prove the conclusion (i). Note $C_{\max}^{LPT}(I) < 3P_{last}^{LPT}(I)$, then $C_{\max}^{LPT}\left(I'\right) < 3P_{last}^{LPT}\left(I'\right)$ immediately holds, which implies $C_{\max}^{*}\left(I'\right) < 3P_{last}^{LPT}\left(I'\right)$, i.e., in the optimal schedule of instance I', each machine processes at most two jobs. Hence, by Lemma 1, LPT schedule is the optimal schedule for I', i.e.,

$$C_{\max}^{LPT}\left(I'\right) = C_{\max}^{*}\left(I'\right).$$

Based on (2) and (4), we have $C_{\max}^{LPT}(I) = C_{\max}^{LPT}\left(I'\right) = C_{\max}^{*}\left(I'\right) \leq C_{\max}^{*}(I)$. Hence,

$$C_{\max}^{LPT}(I) = C_{\max}^{*}(I),$$

which completes the proof of (i).

Next, we prove the conclusion (ii). By contradiction, assuming $C_{\max}^{*}(I) < 3P_{last}^{LPT}(I)$, from (2) and (3), we have

$$C_{\max}^{*}\left(I'\right) < 3P_{last}^{LPT}\left(I'\right).$$

Known from the proof of (i), we also have

$$C_{\max}^{LPT}\left(I'\right) = C_{\max}^{*}\left(I'\right),$$

which implies $C_{\max}^{LPT}\left(I'\right) < 3P_{last}^{LPT}\left(I'\right)$. By (3) and (4), we obtain

$$C_{\max}^{LPT}(I) < 3P_{last}^{LPT}(I),$$

which is a contradiction to the condition in (ii). Therefore, the conclusion (ii) is proved. □

Based on Lemma 2, given an instance $\mathcal{I}_0$ of the multi-agent problem $Pm|CO|C_{\max}^1, C_{\max}^2, \ldots, C_{\max}^g$, all agents can be classified into two sets, denoted by $\mathcal{A}$ and $\overline{\mathcal{A}}$, in the following way.

Definition 2 In instance $\mathcal{I}_0$, for each single-agent instance I_0^i $(i \in \mathcal{G})$, consider the LPT schedule $\sigma^{LPT}(I_0^i)$.

- If machine $M_1\left(\sigma^{LPT}\left(I_0^i\right)\right)$ processes only one job, then $i \in \mathcal{A}$;
- If machine $M_1\left(\sigma^{LPT}\left(I_0^i\right)\right)$ processes at least two jobs, and $C_{\max}^{LPT}\left(I_0^i\right) < 3P_{last}^{LPT}(I_0^i)$, then $i \in \mathcal{A}$;
- If machine $M_1\left(\sigma^{LPT}\left(I_0^i\right)\right)$ processes at least two jobs, and $C_{\max}^{LPT}\left(I_0^i\right) \geq 3P_{last}^{LPT}(I_0^i)$, then $i \in \overline{\mathcal{A}}$.

By the definition of the agent sets $\mathcal{A}$ and $\overline{\mathcal{A}}$, we have $\mathcal{G} = \mathcal{A} \cup \overline{\mathcal{A}}$, and

$$C_{\max}^{*}\left(I_0^i\right) \begin{cases} = C_{\max}^{LPT}\left(I_0^i\right), & i \in \mathcal{A}, \\ \geq 3P_{last}^{LPT}\left(I_0^i\right), & i \in \overline{\mathcal{A}}. \end{cases}$$

3 Adjoint instance $\mathcal{I}$

For each instance $\mathcal{I}_0$, we construct a new instance $\mathcal{I}$ (called the *adjoint instance*) of the multi-agent problem $Pm|CO|C_{\max}^1, C_{\max}^2, \ldots, C_{\max}^g$, and define I^i $(i = 1, 2, \ldots, g)$ as the instance of the single-agent problem $Pm||C_{\max}$ with only jobs of agent i in the adjoint instance $\mathcal{I}$.

The data of $\mathcal{I}$ are the same as that in $\mathcal{I}_0$ except the following modification regarding agents in $\mathcal{A}$. In the instance $\mathcal{I}$, for each

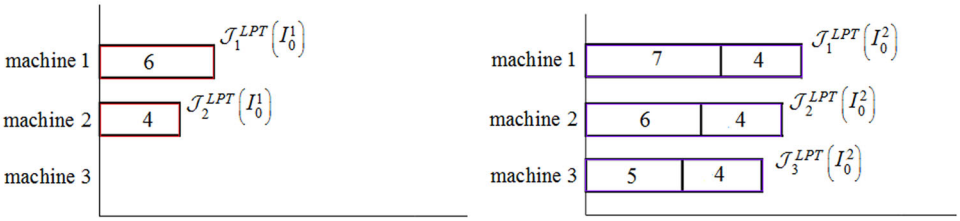

Fig. 1 LPT schedules of I_0^1 and I_0^2 in instance $\mathcal{I}_0$

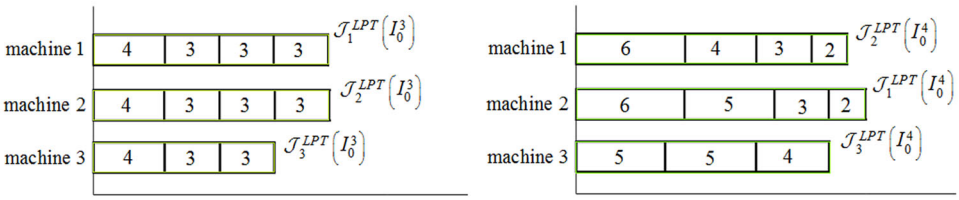

Fig. 2 LPT schedules of I_0^3 and I_0^4 in instance $\mathcal{I}_0$

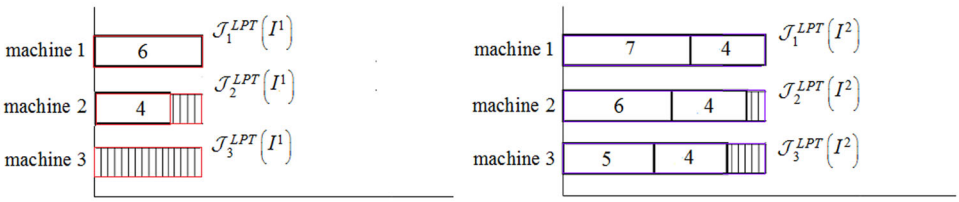

Fig. 3 LPT schedules of I^1 and I^2 in the adjoint instance $\mathcal{I}$

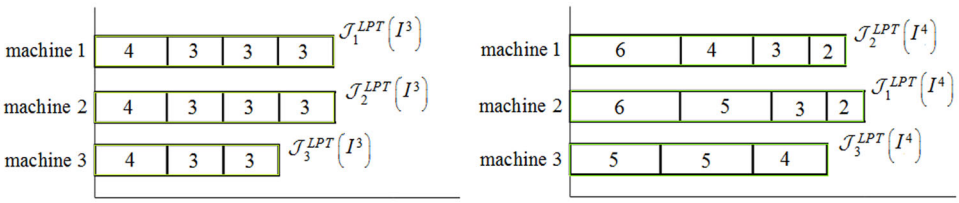

Fig. 4 LPT schedules of I^3 and I^4 in the adjoint instance $\mathcal{I}$

agent $i\,(i \in \mathcal{A})$, some new sufficiently *tiny* jobs are added to agent i such that $C_{\max}^{LPT}\left(I^i\right) = C_{\max}^{LPT}\left(I_0^i\right)$, and each machine has the same processing load in its new LPT schedule. The word *tiny* means the processing time of each added job is very small. We assume each *tiny* job has the same processing time. See Figs. 1 and 3 in the following example.

Example 1 There are four agents in the original instance $\mathcal{I}_0$. Agent 1 has two jobs, agent 2 has six jobs, agent 3 has eleven jobs, and agent 4 also has eleven jobs. The LPT schedules of the four single-agent instances I_0^1, I_0^2, I_0^3 and I_0^4 are given in Figs. 1 and 2.

According to the definition of agent sets $\mathcal{A}$ and $\overline{\mathcal{A}}$, agents 1 and 2 belong to set $\mathcal{A}$, and agents 3 and 4 belong to set $\overline{\mathcal{A}}$. Hence, based on the definition of the adjoint instance $\mathcal{I}$, we could give the corresponding LPT schedules of the four single-agent instances I^1, I^2, I^3 and I^4, which are presented in Figs. 3 and 4.

Remark For each agent i $(i \in \mathcal{A})$, in its LPT schedule $\sigma^{LPT}(I^i)$, the processing loads on all machines are the same, and the machine with no *tiny* job processes the least number of jobs. Known from the definition of $M_k\,(\sigma)$, that machine is the machine $M_1\left(\sigma^{LPT}(I^i)\right)$. The corresponding job set is denoted as $\mathcal{J}_1{}^{LPT}(I^i)$, which includes no *tiny* job. Please see Fig. 3.

With the explanation in Example 1, for the adjoint instance $\mathcal{I}$, we have the following conclusions regarding agents in $\mathcal{A}$ and $\overline{\mathcal{A}}$.

$$C_{\max}^{*}\left(I^i\right) = C_{\max}^{LPT}\left(I^i\right) = \frac{1}{m}\sum_{j=1}^{n_i'} p_j^i,\ i \in \mathcal{A}, \tag{5}$$

$$P_{last}^{LPT}\left(I^i\right) = P_{last}^{LPT}\left(I_0^i\right),\ i \in \mathcal{A}, \tag{6}$$

where n_i' denotes the number of jobs of agent i in the adjoint instance $\mathcal{I}$. On the other side, for agent $i\,(i \in \overline{\mathcal{A}})$, all its data are the same as that in instance $\mathcal{I}_0$. Hence, we have

$$C_{\max}^{*}\left(I^i\right) \geq 3P_{last}^{LPT}(I^i),\ i \in \overline{\mathcal{A}} \tag{7}$$

$$P_{last}^{LPT}\left(I^i\right) = P_{last}^{LPT}\left(I_0^i\right),\ i \in \overline{\mathcal{A}}. \tag{8}$$

Next, we introduce the definition of α point for an agent, based on which an algorithm is proposed for the adjoint instance $\mathcal{I}$.

Definition 3 For each agent i in the adjoint instance $\mathcal{I}$, we define its **α point** as

$$\alpha_i = \begin{cases} \frac{1}{m}\sum_{j=1}^{n_i'} p_j^i, & i \in \mathcal{A} \\ \max\left\{\frac{1}{m}\sum_{j=1}^{n_i'} p_j^i, 3P_{last}^{LPT}(I^i)\right\}, & i \in \overline{\mathcal{A}}. \end{cases}$$

Clearly,

$$\alpha_i \geq \frac{1}{m}\sum_{j=1}^{n_i'} p_j^i, i \in \mathcal{A} \cup \overline{\mathcal{A}} \tag{9}$$

$$\alpha_i \geq 3P_{last}^{LPT}\left(I^i\right), i \in \overline{\mathcal{A}}. \tag{10}$$

Furthermore, by (5) and (7), α_i is a lower bound on the minimum makespan of instance I^i, i.e.,

$$C_{\max}^*\left(I^i\right) \geq \alpha_i, i \in \mathcal{A} \cup \overline{\mathcal{A}}. \tag{11}$$

For ease of argument, assume all agents are numbered according to the non-decreasing α point values in this paper, i.e.,

$$\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_g. \tag{12}$$

In Example 1, $\alpha_1 = 6$, $\alpha_2 = 11$, $\alpha_3 = 12$ and $\alpha_4 = 15$, which satisfy $\alpha_1 \leq \alpha_2 \leq \alpha_3 \leq \alpha_4$.

4 Schedule for adjoint instance $\mathcal{I}$

In this section, an algorithm is proposed for the adjoint instance $\mathcal{I}$, in which agents are assigned to the m machines in the non-decreasing order of α point values, i.e., the increasing order of agents' indexes. Then, the performance of the algorithm is analyzed.

Definition 4 Algorithm LLS (LPT List Scheduling): Denote by $\sigma_0(\mathcal{I})$ the initial schedule, which is an empty schedule. Let $M_k(\sigma_0(\mathcal{I})) = k, k = 1, 2, \ldots, m$. Then

1: **for** each agent $i = 1 : g$ **do**
2: **if** $i \in \mathcal{A}$ **then**
3: **for** each $k = 1 : m$ **do**
4: Arrange all jobs in set $\mathcal{J}_k^{LPT}\left(I^i\right)$ to machine k.
5: **end for**
6: **else**
7: **for** each $k = 1 : m$ **do**
8: Arrange all jobs in set $\mathcal{J}_k^{LPT}\left(I^i\right)$ to machine $M_{m-k+1}(\sigma_{i-1}(\mathcal{I}))$.
9: **end for**
10: **end if**

Table 1 Details of Algorithm LLS for Example 1

Agent 1 ($i \in \mathcal{A}$)	Start job set $\mathcal{J}_k^{LPT}\left(I^1\right)$ ($k = 1, 2, 3$) on machine k at time 0
Agent 2 ($i \in \mathcal{A}$)	Start job set $\mathcal{J}_k^{LPT}\left(I^2\right)$ ($k = 1, 2, 3$) on machine k at time 6
Agent 3 ($i \in \overline{\mathcal{A}}$)	Start job set $\mathcal{J}_1^{LPT}\left(I^3\right)$ on machine $M_3(\sigma_2(\mathcal{I}))$ (i.e., machine 3) at time 17
	Start job set $\mathcal{J}_2^{LPT}\left(I^3\right)$ on machine $M_2(\sigma_2(\mathcal{I}))$ (i.e., machine 2) at time 17
	Start job set $\mathcal{J}_3^{LPT}\left(I^3\right)$ on machine $M_1(\sigma_2(\mathcal{I}))$ (i.e., machine 1) at time 17
Agent 4 ($i \in \overline{\mathcal{A}}$)	Start job set $\mathcal{J}_1^{LPT}\left(I^4\right)$ on machine $M_3(\sigma_3(\mathcal{I}))$ (i.e., machine 1) at time 27
	Start job set $\mathcal{J}_2^{LPT}\left(I^4\right)$ on machine $M_2(\sigma_3(\mathcal{I}))$ (i.e., machine 3) at time 30
	Start job set $\mathcal{J}_3^{LPT}\left(I^4\right)$ on machine $M_1(\sigma_3(\mathcal{I}))$ (i.e., machine 2) at time 30

11: On each machine, jobs of agent i are processed after the jobs of agent 1, agent 2,…, and agent $i-1$. The obtained schedule is denoted by $\sigma_i(\mathcal{I})$.
12: **end for**

Let $\sigma(\mathcal{I})$ denote the final obtained schedule $\sigma_g(\mathcal{I})$, and the schedule generated by Algorithm LLS is referred to as the LLS schedule.

Remark In Algorithm LLS, the operation in the case $i \in \mathcal{A}$ ensures that there is no *tiny* job on machine 1 in the obtained LLS schedule since the set $\mathcal{J}_1^{LPT}\left(I^i\right)$ includes no *tiny* job.

We now return to Example 1. In Example 1, agents 1 and 2 belong to set $\mathcal{A}$, and agents 3 and 4 belong to set $\overline{\mathcal{A}}$. The details of Algorithm LLS are given in Table 1, and the LLS schedule $\sigma(\mathcal{I})$ is presented in Fig. 5.

Lemma 3 *During the process of Algorithm LLS, for each schedule $\sigma_i(\mathcal{I})$, we have*

$$\Delta\left(\sigma_i\left(\mathcal{I}\right)\right) \leq \max_{1\leq j\leq i} \Delta\left(\sigma^{LPT}\left(I^j\right)\right), \quad i = 1, 2, \ldots, g.$$

Proof This lemma is proved by induction on i. When $i = 1$, $\Delta\left(\sigma_1\left(\mathcal{I}\right)\right) = \Delta\left(\sigma^{LPT}\left(I^1\right)\right)$, i.e., the conclusion holds.

Assuming the conclusion holds for schedule $\sigma_i\left(\mathcal{I}\right)(1 \leq i < g)$, then according to Algorithm LLS, schedule $\sigma_{i+1}\left(\mathcal{I}\right)$ satisfies

$$\begin{aligned} \Delta\left(\sigma_{i+1}\left(\mathcal{I}\right)\right) &= \max_{1\leq k<l\leq m} \left|\left(CM_{m+1-k}\left(\sigma_i\left(\mathcal{I}\right)\right) + \mathcal{P}_k^{LPT}\left(I^{i+1}\right)\right)\right. \\ &\quad \left. - \left(CM_{m+1-l}\left(\sigma_i\left(\mathcal{I}\right)\right) + \mathcal{P}_l^{LPT}\left(I^{i+1}\right)\right)\right| \\ &= \max_{1\leq k<l\leq m} \left|\left(CM_{m+1-k}\left(\sigma_i\left(\mathcal{I}\right)\right) - CM_{m+1-l}\left(\sigma_i\left(\mathcal{I}\right)\right)\right)\right. \\ &\quad \left. - \left(\mathcal{P}_l^{LPT}\left(I^{i+1}\right) - \mathcal{P}_k^{LPT}\left(I^{i+1}\right)\right)\right|. \end{aligned}$$

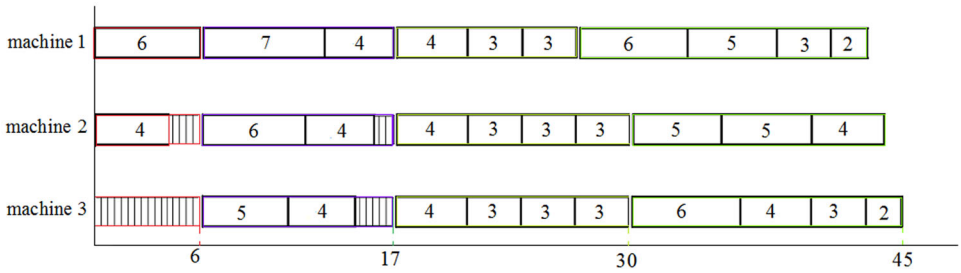

Fig. 5 LLS schedule of the adjoint instance $\mathcal{I}$ in Example 1

Noting $\left(CM_{m+1-k}\left(\sigma_i\left(\mathcal{I}\right)\right)-CM_{m+1-l}\left(\sigma_i\left(\mathcal{I}\right)\right)\right)\cdot\left(\mathcal{P}_l^{LPT}\left(I^{i+1}\right)-\mathcal{P}_k^{LPT}\left(I^{i+1}\right)\right)\geq 0$, we have

$$\begin{aligned}\Delta\left(\sigma_{i+1}\left(\mathcal{I}\right)\right)&\leq\max_{1\leq k<l\leq m}\max\left\{\left|CM_{m+1-k}\left(\sigma_i\left(\mathcal{I}\right)\right)-CM_{m+1-l}\left(\sigma_i\left(\mathcal{I}\right)\right)\right|,\right.\\&\quad\left.\left|\mathcal{P}_l^{LPT}\left(I^{i+1}\right)-\mathcal{P}_k^{LPT}\left(I^{i+1}\right)\right|\right\}\\&=\max\left\{\begin{array}{l}\max\limits_{1\leq k<l\leq m}\left|CM_{m+1-k}\left(\sigma_i\left(\mathcal{I}\right)\right)-CM_{m+1-l}\left(\sigma_i\left(\mathcal{I}\right)\right)\right|,\\\max\limits_{1\leq k<l\leq m}\left|\mathcal{P}_l^{LPT}\left(I^{i+1}\right)-\mathcal{P}_k^{LPT}\left(I^{i+1}\right)\right|\end{array}\right\}\\&=\max\left\{\Delta\left(\sigma_i\left(\mathcal{I}\right)\right),\Delta\left(\sigma^{LPT}\left(I^{i+1}\right)\right)\right\}\\&\leq\max_{1\leq j\leq i+1}\Delta\left(\sigma^{LPT}\left(I^j\right)\right),\end{aligned}$$

where the last inequality holds by induction hypothesis. Hence, the conclusion holds for schedule $\sigma_{i+1}\left(\mathcal{I}\right)$. This completes the proof the lemma. □

Based on Lemma 3, we have the following lemma, which is needed to analyze the performance of Algorithm LLS for the adjoint instance $\mathcal{I}$.

Lemma 4 *During the process of Algorithm LLS, each schedule $\sigma_i\left(\mathcal{I}\right)$ satisfies*

$$\Delta\left(\sigma_i\left(\mathcal{I}\right)\right)\leq\frac{1}{3}\alpha_i,\quad i=1,2,\ldots,g.$$

Proof It is first shown that $\Delta\left(\sigma^{LPT}\left(I^j\right)\right)\leq\frac{1}{3}\alpha_j$, $(j=1,2,\ldots,g)$. The inequality holds trivially if $j\in\mathcal{A}$. If $j\in\overline{\mathcal{A}}$, then by inequalities (1) and (10), we have $\Delta\left(\sigma^{LPT}\left(I^j\right)\right)\leq\frac{1}{3}\alpha_j$.

Hence, from Lemma 3,

$$\Delta\left(\sigma_i\left(\mathcal{I}\right)\right)\leq\max_{1\leq j\leq i}\Delta\left(\sigma^{LPT}\left(I^j\right)\right)\leq\max_{1\leq j\leq i}\frac{1}{3}\alpha_j=\frac{1}{3}\alpha_i,$$

where the last equality holds because all agents are numbered in the non-decreasing order of α point values. This completes the proof of the lemma. □

Lemma 5 *In the schedule $\sigma\left(\mathcal{I}\right)$ generated by Algorithm LLS on the adjoint instance $\mathcal{I}$, the makespan of agent i satisfies the following inequality*

$$C_{\max}^i\left(\sigma\left(\mathcal{I}\right)\right)\leq\left(i+\left(\frac{1}{3}-\frac{1}{3m}\right)\right)C_{\max}^*\left(I_i\right),\ i=1,2,\ldots,g.$$

Proof Since the makespan of agent i is unchanged from schedule $\sigma_i\left(\mathcal{I}\right)$ to $\sigma_g\left(\mathcal{I}\right)$(i.e.,$\sigma\left(\mathcal{I}\right)$), we only need to prove $C_{\max}^i\left(\sigma_i\left(\mathcal{I}\right)\right)\leq\left(i+\left(\frac{1}{3}-\frac{1}{3m}\right)\right)C_{\max}^*\left(I^i\right)$. Note

$$C_{\max}^i\left(\sigma_i\left(\mathcal{I}\right)\right)=CM_1\left(\sigma_i\left(\mathcal{I}\right)\right)=\Delta\left(\sigma_i\left(\mathcal{I}\right)\right)+CM_m\left(\sigma_i\left(\mathcal{I}\right)\right)\tag{13}$$

$$\leq\Delta\left(\sigma_i\left(\mathcal{I}\right)\right)+\frac{1}{m}\left(\sum_{k=1}^{i}\sum_{j=1}^{n_i'}p_j^k-\Delta\left(\sigma_i\left(\mathcal{I}\right)\right)\right)\tag{14}$$

$$=\frac{1}{m}\sum_{k=1}^{i}\sum_{j=1}^{n_i'}p_j^k+\left(1-\frac{1}{m}\right)\Delta\left(\sigma_i\left(\mathcal{I}\right)\right)\tag{15}$$

$$\leq\sum_{k=1}^{i}\alpha_k+\frac{1}{3}\left(1-\frac{1}{m}\right)\alpha_i\tag{16}$$

$$\leq\left(i+\frac{1}{3}\left(1-\frac{1}{m}\right)\right)\alpha_i\tag{17}$$

$$\leq\left(i+\left(\frac{1}{3}-\frac{1}{3m}\right)\right)C_{\max}^*\left(I^i\right),\tag{18}$$

where the second inequality holds by (9) and Lemma 4, and the final inequality holds by (11). Hence, the proof is completed. □

5 Schedule for instance $\mathcal{I}_0$

In this section, based on Algorithm LLS, we firstly introduce a new approximation algorithm for the original instance $\mathcal{I}_0$ of problem $Pm|CO|C_{\max}^1, C_{\max}^2, \ldots, C_{\max}^g$ and then analyze the performance of the algorithm. Finally, an instance is given to show the performance analysis is tight.

Definition 5 Algorithm GLLS (Generalized LPT List Scheduling): For the instance $\mathcal{I}_0$ of problem $Pm|CO|C_{\max}^1, C_{\max}^2, \ldots, C_{\max}^g$,

Step 1: Construct the adjoint instance $\mathcal{I}$ by the method introduced in Sect. 3;
Step 2: By Algorithm LLS, generate the LLS schedule $\sigma\left(\mathcal{I}\right)$ of instance $\mathcal{I}$;

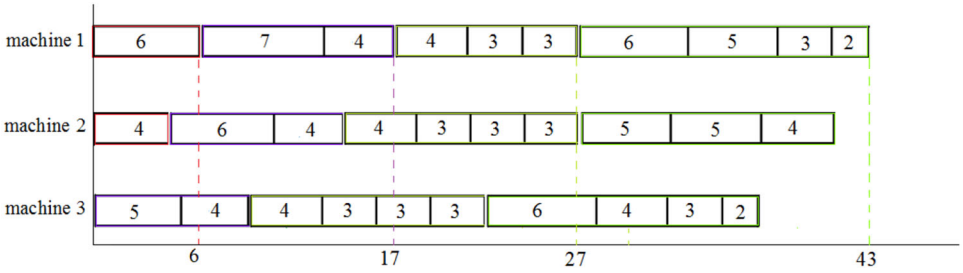

Fig. 6 GLLS schedule of the original instance $\mathcal{I}_0$ in Example 1

Step 3: In the LLS schedule $\sigma(\mathcal{I})$, remove all the added tiny jobs from each machine, and the remaining jobs step forward, i.e., on each machine, jobs of an agent are started as soon as the jobs of previous agents are completed.

Denote by $\sigma(\mathcal{I}_0)$ the final obtained schedule for instance $\mathcal{I}_0$, and the schedule generated by Algorithm GLLS is referred to as the GLLS schedule.

For Example 1, the GLLS schedule $\sigma(\mathcal{I}_0)$ is presented in Fig. 6.

For the GLLS schedule $\sigma(\mathcal{I}_0)$, two main conclusions are introduced in the following two lemmas.

Lemma 6 *In the schedule $\sigma(\mathcal{I}_0)$ generated by Algorithm GLLS on instance $\mathcal{I}_0$, agent i is the ith completed agent, i.e., we have*

$$C^i_{\max}(\sigma(\mathcal{I}_0)) < C^{i+1}_{\max}(\sigma(\mathcal{I}_0)), i = 1, 2, \ldots, g-1.$$

Proof The conclusion holds trivially if $i+1 \in \overline{\mathcal{A}}$ because each set $\mathcal{J}_k^{LPT}\left(\mathcal{I}_0^{i+1}\right)$ $(k = 1, 2, \ldots, m)$ is nonempty.

If $i+1 \in \mathcal{A}$, then the makespan of agent $i+1$ in the GLLS schedule $\sigma(\mathcal{I}_0)$ satisfies

$$C^{i+1}_{\max}(\sigma(\mathcal{I}_0)) \geq C^{i+1}_1(\sigma(\mathcal{I}_0)),$$

where $C^{i+1}_1(\sigma(\mathcal{I}_0))$ denotes the completion time of agent $i+1$ on machine 1 in the schedule $\sigma(\mathcal{I}_0)$. Since there is no *tiny* jobs on machine 1 in the schedule $\sigma(\mathcal{I})$ (see the *Remark* after Definition 4), this implies that $C^{i+1}_1(\sigma(\mathcal{I}_0)) = C^{i+1}_1(\sigma(\mathcal{I}))$. Hence,

$$C^{i+1}_{\max}(\sigma(\mathcal{I}_0)) \geq C^{i+1}_1(\sigma(\mathcal{I})). \tag{19}$$

Next, we will show $C^{i+1}_1(\sigma(\mathcal{I})) > C^i_{\max}(\sigma(\mathcal{I}_0))$. In the schedule $\sigma_i(\mathcal{I})$,

$$\begin{aligned} C^i_{\max}(\sigma_i(\mathcal{I})) &= CM_1(\sigma_i(\mathcal{I})) = CM_m(\sigma_i(\mathcal{I})) + \Delta(\sigma_i(\mathcal{I})) \\ &\leq CM_m(\sigma_i(\mathcal{I})) + \frac{1}{3}\alpha_i \\ &< CM_m(\sigma_i(\mathcal{I})) + \alpha_i \\ &\leq CM_m(\sigma_i(\mathcal{I})) + \alpha_{i+1} \\ &\leq C^i_1(\sigma_i(\mathcal{I})) + \alpha_{i+1} \\ &= C^{i+1}_1(\sigma_{i+1}(\mathcal{I})) = C^{i+1}_1(\sigma(\mathcal{I})), \end{aligned}$$

where the first inequality holds by Lemma 4. Since $C^i_{\max}(\sigma(\mathcal{I}_0)) \leq C^i_{\max}(\sigma_i(\mathcal{I}))$, we have

$$C^{i+1}_1(\sigma(\mathcal{I})) > C^i_{\max}(\sigma(\mathcal{I}_0)). \tag{20}$$

Based on (19) and (20), we obtain

$$C^{i+1}_{\max}(\sigma(\mathcal{I}_0)) > C^i_{\max}(\sigma(\mathcal{I}_0)), \tag{21}$$

i.e., the conclusion holds in the case $i+1 \in \mathcal{A}$, which completes the induction and the proof of this lemma. □

Lemma 7 *In the schedule $\sigma(\mathcal{I}_0)$ generated by Algorithm GLLS on instance $\mathcal{I}_0$, the makespan of agent i satisfies the following inequality*

$$C^i_{\max}(\sigma(\mathcal{I}_0)) \leq \left(i + \left(\frac{1}{3} - \frac{1}{3m}\right)\right) C^*_{\max}\left(I_0^i\right), i = 1, 2, \ldots, g.$$

Proof Known from the construction method of the adjoint instance $\mathcal{I}$ in Sect. 3, for each agent i,

$$C^*_{\max}\left(I^i\right) = C^*_{\max}\left(I_0^i\right), i = 1, 2, \ldots, g.$$

By Lemma 5, we know

$$C^i_{\max}(\sigma(\mathcal{I})) \leq \left(i + \left(\frac{1}{3} - \frac{1}{3m}\right)\right) C^*_{\max}\left(I^i\right), i = 1, 2, \ldots, g.$$

Furthermore, according to Step 3 in Algorithm GLLS,

$$C^i_{\max}(\sigma(\mathcal{I}_0)) \leq C^i_{\max}(\sigma(\mathcal{I})), i = 1, 2, \ldots, g.$$

From the three inequalities above, we obtain

$$C^i_{\max}(\sigma(\mathcal{I}_0)) \leq \left(i + \left(\frac{1}{3} - \frac{1}{3m}\right)\right) C^*_{\max}\left(I_0^i\right), i = 1, 2, \ldots, g,$$

which completes the proof of this lemma. □

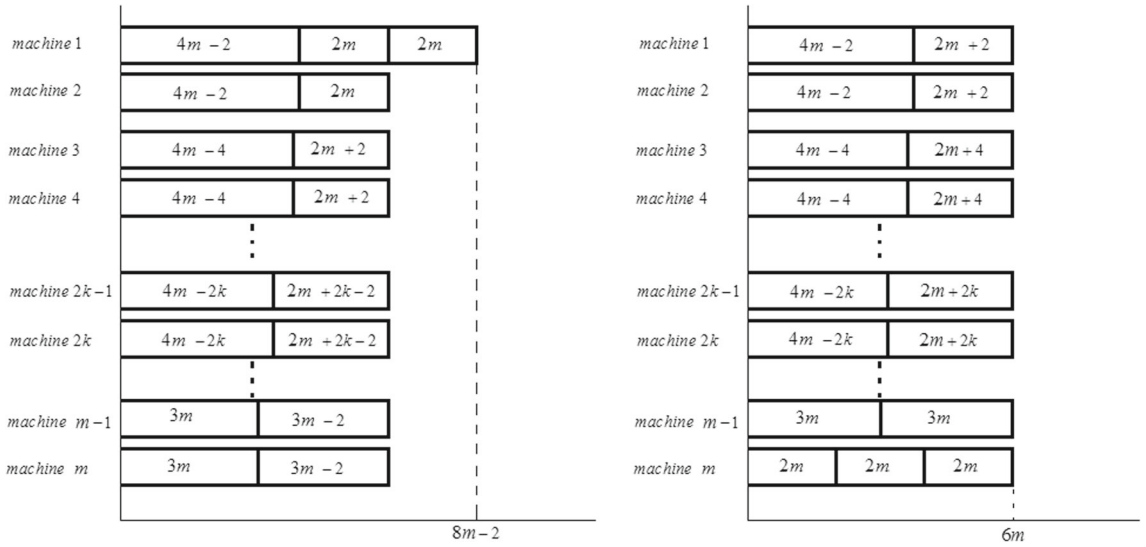

Fig. 7 The LPT schedule and the optimal schedule of I_0^1 if m is an even number

With Lemmas 6 and 7, the following conclusion holds immediately.

Theorem 8 *In the schedule $\sigma(\mathcal{I}_0)$ generated by Algorithm GLLS on instance $\mathcal{I}_0$, agent i $(1 \le i \le g)$ is the ith completed agent, and its completion time is at most $i + \left(\frac{1}{3} - \frac{1}{3m}\right)$ times its minimum makespan, i.e., the performance ratio vector of Algorithm GLLS is*

$$\left(1+\left(\frac{1}{3}-\frac{1}{3m}\right), 2+\left(\frac{1}{3}-\frac{1}{3m}\right), \ldots, g+\left(\frac{1}{3}-\frac{1}{3m}\right)\right).$$

In the rest of this section, we give an instance to show the performance ratio vector of Algorithm GLLS is tight. In instance $\mathcal{I}_0$, agent 1 has $2m+1$ jobs, and the processing times of jobs of agent 1 are given as follows.

- $p_{2i-1}^1 = 4m - 2i$, $p_{2i}^1 = 4m - 2i$, $i = 1, \ldots, m$,
- $p_{2m+1}^1 = 2m$.

If m is an even number, the LPT schedule and the optimal schedule of instance I_0^1 are presented in Fig. 7.

If m is an odd number, the LPT schedule and the optimal schedule of instance I_0^1 are presented in Fig. 8.

Therefore, either m is an even number or not, regarding agent 1, we could always construct instance I_0^1 such that agent 1 belongs to $\overline{\mathcal{A}}$, and

$$C_{\max}^*\left(I_0^1\right) = 6m, C_{\max}^{LPT}\left(I_0^1\right) = 8m - 2, \alpha_1 = 6m.$$

For each agent $i (i = 2, 3, \ldots, g)$, let $p_j^i = 6m + \varepsilon$ $(j = 1, 2, \ldots, m)$. Hence, these agents belong to $\mathcal{A}$, and

$$C_{\max}^{LPT}(I_0^i) = C_{\max}^*(I_0^i) = \alpha_i = 6m + \varepsilon.$$

In the constructed instance $\mathcal{I}_0$, we have $\alpha_1 \le \alpha_2 \le \cdots \le \alpha_g$, and in the GLLS schedule on the instance $\mathcal{I}_0$, agent $i (i = 1, 2, \ldots, g)$ is the ith completed agent. Moreover, the makespan of agent i satisfies

$$C_{\max}^i(\sigma(\mathcal{I}_0)) = i \cdot 6m + 2m - 2 + (i-1)\varepsilon, i = 1, 2, \ldots, g.$$

Therefore,

$$\frac{C_{\max}^1(\sigma(\mathcal{I}_0))}{C_{\max}^*\left(I_0^1\right)} = 1 + \left(\frac{1}{3} - \frac{1}{3m}\right) \tag{22}$$

$$\lim_{\varepsilon \to 0} \frac{C_{\max}^i(\sigma(\mathcal{I}_0))}{C_{\max}^*\left(I_0^i\right)} = \lim_{\varepsilon \to 0} \frac{i \cdot 6m + 2m - 2 + (i-1)\varepsilon}{6m + \varepsilon} = i + \left(\frac{1}{3} - \frac{1}{3m}\right), i = 2, 3, \ldots, g \tag{23}$$

which implies the performance ratio vector is tight.

6 Concluding remarks

This paper considers the multi-agent scheduling problem on m identical parallel machines. Each agent has a set of non-preemptive jobs and each agent competes to minimize its own makespan. In order to solve the problem, a lower bound on the minimum makespan of each agent (i.e., the α point value) is introduced. Then, we propose an algorithm in which jobs of all agents are arranged to be processed on machines according to the non-decreasing order of the α point values. In the obtained schedule, we prove the agent with the ith smallest α point value is the ith completed agent, and its completion time is at most $i + \left(\frac{1}{3} - \frac{1}{3m}\right)$ times its minimum

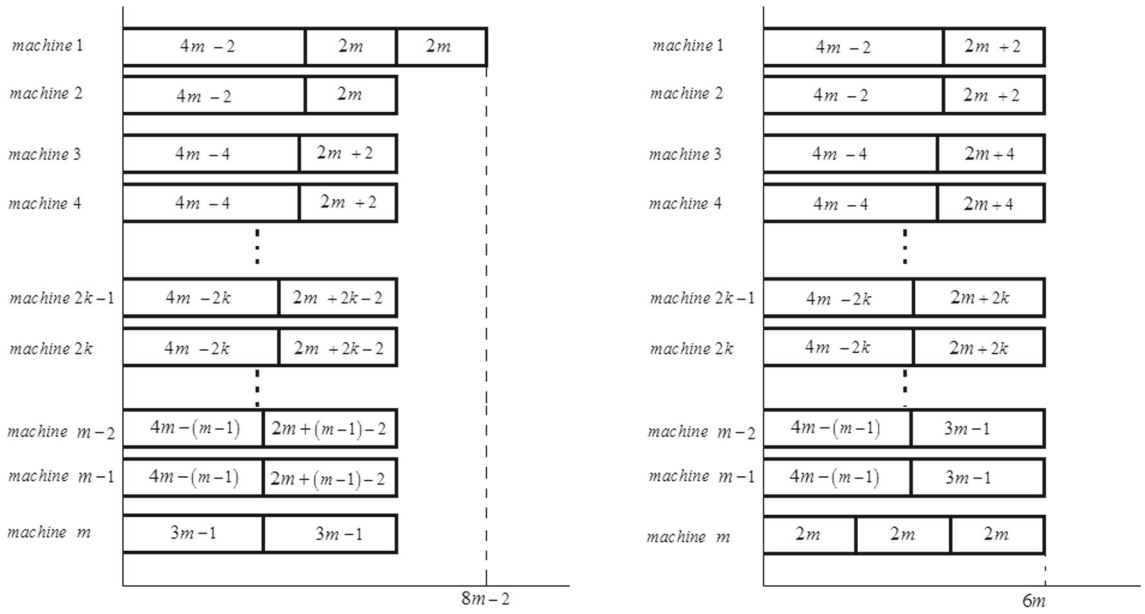

Fig. 8 The LPT schedule and the optimal schedule of I_0^1 if m is an odd number

makespan. Finally, an instance is constructed to show the performance analysis is tight.

There are two limitations regarding the result in this paper, which could be improved in the future research.

- The performance ratio vector in this paper is different from that introduced by Lee et al. (2009). In the paper Lee et al. (2009), the worst-case performance ratio of each agent could be determined by the user of the algorithm when the relationship among the worst-case performance ratios of all agents are satisfied. By the algorithm proposed in our paper, the scheduler could only know the worst-case performance ratio of the agent with the ith smallest α point value. In the future work, we may employ the concept of approximation in Lee et al. (2009) and try to propose a similar algorithm.
- In this paper, jobs of each agent $k (k \in \mathcal{G})$ are divided into m job sets (i.e., $\mathcal{J}_1^{LPT}(I^i), \mathcal{J}_2^{LPT}(I^i), \ldots, \mathcal{J}_m^{LPT}(I^i)$). The m job sets are assigned to the m machines, respectively, such that the agent is completed as early as possible, and the assignment is carried out in the non-decreasing order of the α point values. In the final obtained schedule, we show the worst-case performance ratio of the ith completed agent is $i + \left(\frac{1}{3} - \frac{1}{3m}\right)$, where $\frac{1}{3} - \frac{1}{3m}$ is the relative error of the LPT rule for the single-agent problem.
  The approach is not applicable for the problem with uniform machines. For example, let us consider the problem with two machines. By the approach introduced in this paper, given each agent, the two job sets are assigned to the two machines, respectively, such that the agent is completed as early as possible. Therefore, if the two machines are identical, then the profile of the obtained schedule is at most the minimal value of the two profiles of the previous partial schedule and the LPT schedule of the single-agent instance. The profile of a schedule is the difference between the completion times of the last two completed jobs, respectively, on the two machines (Pinedo 2011). If the two machines are uniform, the profile of the obtained schedule may be the sum of the two profiles of the previous partial schedule and the LPT schedule of the single-agent instance, which implies the relative error may increase and be larger than that of the LPT rule for the single-agent problem $Q2||C_{\max}$. In some special cases, the relative error even goes to infinity.
  Hence, as an extension of this paper, the multi-agent problem with uniform machines could be studied. We may try to propose a different approach in which jobs of each agent are assigned to machines one by one, instead of assigning the m job sets to the m machines, respectively.

Acknowledgements We are grateful to the referees for providing comments for our paper. The authors thank Dr. Kejun Zhao for reviewing the paper before its formally submission. This work is supported by National Natural Science Foundation of China (Grant Nos. 11201282, 61304209, and 11371137), Innovation Program of Shanghai Municipal Education Commission (Grant No. 14YZ127), Humanities and Social Sciences planning fund of Ministry of Education (Grant No. 17YJAZH024), Pre-research project for young teachers from SUFE, and National project follow-up research project from SUFE.

References

Agnetis, A., Billaut, J. C., Gawiejnowicz, S., Pacciarelli, D., & Soukhal, A. (2014). *Multiagent scheduling-models and algorithms*. Berlin: Springer. ISBN 978-3-642-41879-2.

Agnetis, A., Mirchandani, P. B., Pacciarelli, D., & Pacifici, A. (2004). Scheduling problem with two competing agents. *Operations Research*, *52*, 229–242.

Agnetis, A., Pacciarelli, D., & Pacifici, A. (2007). Multi-agent single machine scheduling. *Annals of Operations Research*, *150*, 3–15.

Agnetis, A., Pascale, G., & Pacciarelli, D. (2009). A Lagrangian approach to single-machine scheduling problems with two competing agents. *Journal of Scheduling*, *12*, 401–415.

Cheng, T. C. E., Ng, C. T., & Yuan, J. J. (2006). Multi-agent scheduling on a single machine to minimize total weighted number of tardy jobs. *Theoretical Computer Science*, *362*, 273–281.

Cheng, T. C. E., Ng, C. T., & Yuan, J. J. (2008). Multi-agent scheduling on a single machine with max-form criteria. *European Journal of Operational Research*, *188*, 603–609.

Fan, B. Q., Cheng, T. C. E., Li, S. S., & Feng, Q. (2013). Bounded parallel-batching scheduling with two competing agents. *Journal of Scheduling*, *16*, 261–271.

Lee, K., Choi, B.-C., Leung, J. Y.-T., & Pinedo, M. L. (2009). Approximation algorithms for multi-agent scheduling to minimize total weighted completion time. *Information Processing Letters*, *109*, 913–917.

Leung, J. Y.-T., Pinedo, M., & Wan, G. (2010). Competitive two-agent scheduling and its applications. *Operations Research*, *58*, 458–469.

Li, S., & Yuan, J. J. (2012). Unbounded parallel-batching scheduling with two competitive agents. *Journal of Scheduling*, *15*, 629–640.

Pinedo, M. L. (2011). *Scheduling-theory, algorithm, and systems* (4th ed.). New York: Springer.

Saule, E., & Trystram, D. (2009). Multi-users scheduling in parallel systems. In *Proceedings of IEEE international parallel and distributed processing symposium 2009, Roma, Italy* (pp. 1–9).

Zhao, K., Lu, X., & Gu, M. (2016). A new approximation algorithm for multi-agent scheduling to minimize makespan on two machines. *Journal of Scheduling*, *19*, 21–31.