Journal of Combinatorial Optimization (2020) 40:333–365
https://doi.org/10.1007/s10878-020-00592-2

Coordination mechanisms for scheduling selfish jobs with favorite machines

Cong Chen[1] · Yinfeng Xu[2]

Published online: 10 June 2020

Abstract
This paper studies the *favorite* machine model, where each machine has different speed for different types of jobs. The model is a natural generalization of the two related machines model and captures the features of some real life problems, such as the CPU–GPU task scheduling, the two products scheduling and the cloud computing task scheduling. We are interested in the game-theoretic version of the scheduling problem in which jobs correspond to *self-interested* users and machines correspond to resources. The goal is to design coordination mechanisms (local policies) with a small price of anarchy (PoA) for the scheduling game of favorite machines. We first analyze the well known Makespan policy for our problem, and provide exact bounds on both the PoA and the strong PoA (SPoA). We also propose a new local policy, called FF-LPT, which outperforms several classical policies (e.g., LPT, SPT, FF-SPT and Makespan) in terms of the PoA, and guarantees fast convergence to a pure Nash equilibrium. Moreover, computational results show that the FF-LPT policy also dominates other policies for random instances, and reveal some insights for practical applications.

Keywords (Strong) price of anarchy · Load balancing game · Coordination mechanism · Related machines

Part of this work was done while the first author was visiting Department of Computer Science, ETH Zurich, Switzerland. A preliminary version of some results of this paper appeared in Proc. COCOA 2017.

✉ Cong Chen
chencong@scut.edu.cn

[1] School of Business Administration, South China University of Technology, Guangzhou, China

[2] School of Management, Xi'an Jiaotong University, Xi'an, China

1 Introduction

The problem of scheduling jobs on *related* machines describes the situation where each machine has a fixed speed for processing all the jobs. However, in many practical situations, one machine may have different speeds for different types of jobs. For example, in the CPU–GPU tasks scheduling problem (Chen et al. 2014), GPU can handle some tasks (e.g., video processing, image analysis, signal processing, etc.) more efficiently than CPU, but CPU is more suitable for a wide range of other tasks. The same situation also appears in some manufacturing factories, such as the production of spare parts for cars (Vakhania et al. 2014). Similarly, in cloud computing, due to a long distance link between a sever and a request, the issue of data latency may occur. Consequently, a task can be processed faster by a nearby server, and different tasks may have different nearby servers (Chen et al. 2020).

Therefore, in this paper, we consider the *favorite* machine model, where each job has a certain set of favorite machines with the shortest processing time and other machines with longer processing times. Specifically, we focus on the *game-theoretic* version of the problem (i.e., load balancing game), where jobs correspond to *self-interested* users who choose which machine to use accordingly without any centralized control, and naturally aim at minimizing their *own cost*. This setting is common in several practical applications. For example, in the decentralized systems, the users have the freedom to choose severs by their own. The selfish behaviors of the players will result in some *equilibrium* in which no player has an incentive to deviate, though the resulting schedule is not necessarily the optimal in terms of the social cost, the makespan. The *price of anarchy* (PoA) (Koutsoupias and Papadimitriou 1999) is a central concept to quantify such inefficiency of Nash equilibria in games. However, several works pointed out that the PoA may be too pessimistic because, even for two unrelated machines and two jobs, the PoA is *unbounded* due to a unnatural Nash equilibrium (see Example 1 below).

Example 1 (*Bad equilibrium for two unrelated machines*) Consider two jobs and two unrelated machines, where the processing times are given by the following table:

	job 1	job 2
machine 1	1	s
machine 2	s	1

The allocation represented by the gray box is a pure Nash equilibrium: if a job moves to the other machine, its own cost increases from s to $s + 1$, where $s \geq 1$. As the optimal makespan is 1 (swap the allocation), even for two machines the ratio between the cost of the worst equilibrium and the optimum is unbounded (at least s).

Some other approaches thus arise to reduce the PoA. One direction is to consider the PoA under other equilibrium concepts than Nash equilibrium, e.g., strong equilibrium (Aumann 1959; Andelman et al. 2009; Fiat et al. 2007), sequential equilibrium (de Jong and Uetz 2014; Hassin and Yovel 2015; Giessler et al. 2016), etc. These equilibrium concepts can somewhat avoid "strange" equilibria, which barely happen in real world,

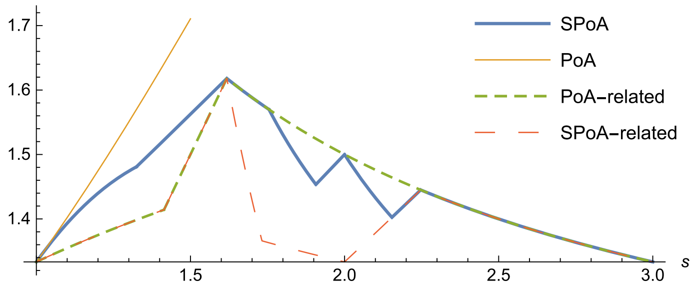


Fig. 1 Comparison between PoA and SPoA in our model and in two related machines (Epstein 2010) (Color figure online)

and hence lower the PoA. Another important direction is to allow a central authority to design protocols and define rewarding rules and hope that the independent and selfish choices of the users, given the rules of the protocols, result in a socially desired outcome. Particularly, the approach, termed *coordination mechanism* (Christodoulou et al. 2009), has received considerable attention in the recent literature (Zhang et al. 2019; Caragiannis et al. 2017; Chen et al. 2017; Azar et al. 2015; Christodoulou et al. 2009; Immorlica et al. 2009; Caragiannis 2013; Chen and Gürel 2012). A coordination mechanism for a scheduling game is a set of *local policies*, one for each machine, that determines how to schedule jobs assigned to that machine. A machine's policy is a function only of the jobs assigned to that machine. This allows the policy to be implemented in a completely distributed and local fashion. Notice that in the problem of coordination mechanism design, the cost of a job is the completion time of the job, which is different from the classical load balancing game. In some literature, the load balancing game is also called the *Makespan policy*, which process all jobs on the same machine in parallel, and so the completion time of a job on a machine is the load of the machine. For clearness, we use the term Makespan policy to represent the load balancing game in the following.

In this paper, we study the favorite machine model for two machines in the above two approaches: We first analyze the PoA for strong equilibria and Nash equilibria of the Makespan policy, respectively; Then we design a new local policy for the problem to further reduce the PoA.

1.1 Our results

In Sect. 3, we provide exact bounds on the strong PoA (i.e., SPoA) of the Makespan policy for all values of s. By the same technique, we also give exact bounds on the PoA. These results extend the case of two related machines with speed ratio s (Epstein 2010), and allow us to compare with the bounds for two related machines (see Fig. 1). Unlike for the two related machines model, the PoA grows with s in our model, and thus the influence of coalitions in strong equilibria is more evident. Note that the worst case of SPoA ($= \frac{\sqrt{5}+1}{2} \approx 1.618$) is attained when $s = \frac{\sqrt{5}+1}{2}$, which coincides with the two related machines case (Epstein 2010). Also, for sufficiently large s, the two problems have the exact same SPoA, though the PoA is very much different.

In Sect. 4, we design a new local policy, termed FF-LPT, for which the PoA and SPoA are both $\frac{4}{3}$. Our results show that the FF-LPT policy outperforms many classical local ordering policies. Specifically, the widely used LPT policy has no bounded PoA for our problem, which can be indicated by the instance of Example 1. The PoA for another classical policy, namely, the SPT policy, is no better than 1.618, since it is already no better than 1.618 for the two related machines model (Cho and Sahni 1980). Similarly, the variation of the SPT policy, FF-SPT policy (formally defined in Sect. 5), also has the PoA no better than 1.618. For the Makespan policy, we have already showed that PoA is unbounded and SPoA is 1.618. Therefore, our FF-LPT policy has a better performance than all the above policies. In addition, we also prove that the players converge to a pure Nash equilibrium in at most n rounds of best-response moves under the FF-LPT policy.

We also evaluate the FF-LPT policy by numerical simulations in Sect. 5. The experimental results show that the FF-LPT policy outperforms the other local ordering policies in terms of the average cases, which supports our theoretical results. The results also reveal some insights for practical problems: (1) The FF-LPT policy tops other policies in both theory (worst case) and numerical results (average case). Hence, the policy could be a good choose for practical problems. (2) When s is small (i.e., close to identical machines), the simple LPT policy has similar performance to our FF-LPT policy. But when s gets larger, the LPT policy can be very bad, and the measure of processing the more efficient jobs primarily (like the FF-LPT policy) can significantly improve the equilibrium. (3) In the perspective of the convergence time to a equilibrium, the Makespan policy is faster than the other policies. In a sense, the Makespan policy can somewhat reduce the number of strategic behaviors of the users.

1.2 Related work

Here we introduce some related results for our work. A summary of the results are shown in Table 1. The Makespan policy (i.e., the load balancing game) has been widely studied. For the most general model, the *unrelated* machines, the PoA is unbounded (Awerbuch et al. 2006) and the SPoA is exactly m (i.e., the number of machines) (Andelman et al. 2009; Fiat et al. 2007). Hence, several restrictions of the unrelated machine model drew more attention from researchers. One of the most studied restrictions is the *related* machine model, in which the processing time of a job on a machine is the size of the job divided by the speed of the machine. For the *related* machine model, PoA $= \Theta(\frac{\log m}{\log \log m})$ (Czumaj and Vöcking 2007) and SPoA $= \Theta(\frac{\log m}{(\log \log m)^2})$ (Fiat et al. 2007). The case of a *small number of machines* is of particular interest. For *two* and *three* machines, PoA $= \frac{\sqrt{5}+1}{2}$ and PoA $= 2$ (Feldmann et al. 2003), respectively. For *two* machines, exact bounds as a function of the *speed ratio* s on both PoA and SPoA were given by Epstein (2010) (see Fig. 1 for the graph of the following functions):

Table 1 A summary of the related results for some classical policies

	SPoA (Makespan)	PoA (Makespan)	SPT	LPT	Best known
Identical	$\frac{2m}{m+1}$	$\frac{2m}{m+1}$	$2 - \frac{1}{m}$	$\frac{4}{3} - \frac{1}{3m}$	$\frac{4}{3} - \frac{1}{3m}$
Related	$\Theta(\frac{\log m}{(\log\log m)^2})$	$\Theta(\frac{\log m}{\log\log m})$	$\Theta(\log m)$	$[1.52, 1.59]$	1.59
$m = 2$	$\frac{\sqrt{5}+1}{2} \simeq 1.618$	1.618	1.618	$\frac{\sqrt{17}+1}{4} \simeq 1.281$	1.281
Unrelated	m	∞	$\Theta(m)$	∞	$\Theta(\log m)$
Favorite	$\Theta(m/k)$	∞	–	–	–
$m = 2$	1.618^*	∞^*	≥ 1.618	∞	$\frac{4}{3}^*$

The results marked by * are proved in this paper

$$\text{PoA} = \begin{cases} 1+\frac{s}{s+2}, & 1 \le s \le \sqrt{2} \\ s, & \sqrt{2} \le s \le \frac{1+\sqrt{5}}{2} \approx 1.618 \\ 1+\frac{1}{s}, & s \ge \frac{1+\sqrt{5}}{2}, \end{cases}$$

$$\text{SPoA} = \begin{cases} 1+\frac{s}{s+2}, & 1 \le s \le \sqrt{2} \\ s, & \sqrt{2} \le s \le \frac{1+\sqrt{5}}{2} \approx 1.618 \\ \frac{1}{s-1}, & \frac{1+\sqrt{5}}{2} \le s \le \sqrt{3} \\ 1+\frac{1}{s+1}, & \sqrt{3} \le s \le 2 \\ \frac{s^2}{2s-1}, & 2 \le s \le s_1 \approx 2.247 \\ 1+\frac{1}{s}, & s \ge s_1. \end{cases}$$

For the *identical* machines model, the PoA and the SPoA for pure equilibria are *identical* and bounded by a *constant*: PoA = SPoA = $\frac{2m}{m+1}$ (Andelman et al. 2009), where the upper and lower bounds on PoA can be deduced from Finn and Horowitz (1979) and Schuurman and Vredeveld (2007), respectively.

Coordination mechanism was first introduced by Christodoulou et al. (2009). In that paper, they first studied the 2 identical machines case and proposed the increasing-decreasing policy with PoA of $\frac{4}{3}$. Then they proved that the PoA of LPT policy for m identical machines is $\frac{4}{3} - \frac{1}{3m}$. Immorlica et al. (2009) studied four classical policies (i.e., Makespan, SPT, LPT and Randomized) for four scheduling model (i.e., identical machines, related machines, unrelated machines, restrict assignment) and surveyed the results for those problems. Particularly, for the two related machines, the PoA for the SPT policy is $\frac{\sqrt{5}+1}{2} \simeq 1.618$ (Cho and Sahni 1980), and the PoA for the LPT policy is $\frac{\sqrt{17}+1}{4} \simeq 1.281$ (Gonzalez et al. 1977). For the unrelated machines model, Azar et al. (2015) designed a non-preemptive local ordering policy for which the PoA is $\Theta(\log m)$, and showed that the ratio is the best possible for any local ordering policy. However, this policy may induce games without pure Nash equilibria. They also presented a technique that transforms this policy to a preemptive one that induces potential games with PoA $O(\log^2 m)$. In this policy, the players converge to a pure Nash equilibrium in n rounds of best-response moves. The result of the preemptive version was improved by Caragiannis (2013) and Caragiannis and Fanelli (2019), by a policy for which the PoA is $O(\log m)$.

The favorite machine model was first introduced by Chen et al. (2020) in an online version. We then studied the game-theoretic version of the model for two machines case. Following our paper, the results of SPoA for Makespan policy was extended to m machines case soon by Chen and Xu (2019). The paper obtained that the SPoA is $\Theta(\frac{m}{\log m})$ for a special case where each job has only one favorite machine, and the SPoA is $\Theta(\frac{m}{k})$ for a general case where each job has at least k favorite machines.

Some studies also focused on the other settings of scheduling game. Hoeksma and Uetz (2019) and Zhang et al. (2019) considered the related machines scheduling problem under the case where social cost is total completion time. Nong et al. (2017) studied a coordination mechanism for a scheduling game with parallel-batching machines Chen et al. (2017) considered scheduling games with proportional deterioration. A two-stage scheduling games was investigated by Ye et al. (2019). For other results of mechanism design for machine scheduling problems, we refer the readers to the survey by Kress et al. (2018) and references therein.

2 Preliminaries

2.1 Model (favorite machines) and basic definitions

The *favorite* machine model of two machines is defined as follows: There are 2 machines to process n jobs, denoted by $M = \{1, 2\}$ and $J = \{1, 2, \ldots, n\}$, respectively. Each job j consists of a pair (s_j, f_j), where s_j is the *size* of job j and f_j is the *favorite machine* of this job. Denote the processing time of job j on machine i by p_{ij}. For a common parameter $s \geq 1$, the processing time of a job on a favorite machine is just its size ($p_{ij} = s_j$ if $i = f_j$), while on a non-favorite machine it is s *times slower* ($p_{ij} = s \cdot s_j$ if $i \neq f_j$). A schedule is an assignment of each job to some machine. The *load* of a machine i is the sum of the processing times of the jobs assigned to machine i. The *makespan* of a schedule is the maximum load over all machines. In case of a tie, without loss of generality we assume that a job with a lower index has a higher priority.

In the *load balancing game* (a.k.a. the Makespan policy), it is often assumed that all jobs on the same machine are processed in parallel, and so the cost of each job is the load of the machine it chooses. All the selfish jobs make their chooses at the same time while minimizing their own costs. The selfish behaviors of the jobs will result in some allocation $x = (x_1, \ldots, x_n)$, where x_j denotes the machine chosen by job j. Let $\ell_i(x) = \sum_{j:x_j=i} p_{ij}$ be the load of machine i. We say that x is a *Nash equilibrium (NE)* if no player j can unilaterally deviate and improve his/her own cost, i.e., move to a machine $\hat{x}_j$ such that $C_j(\hat{x}) < C_j(x)$ where $\hat{x} = (x_1, \ldots, \hat{x}_j, \ldots, x_n)$ is the allocation resulting from j's move and $C_j(x)$ is the cost of job j under allocation x. In a *strong Nash equilibrium (SE)*, we require that in any group of deviating players, at least one of them does not improve: allocation x is a SE if, for any $\hat{x}$ which differs in exactly a subset Γ of players, there is one $j \in \Gamma$ such that $C_j(\hat{x}) \geq C_j(x)$. Since the resulting allocation is not necessarily the optimal in terms of makespan (the social cost), the notion of PoA (and SPoA) is proposed to measure the efficiency of equilibria. The PoA is the worst-case ratio between the social cost (i.e., makespan) of a NE and the optimum: PoA $= \max_{x \in \mathsf{NE}} \frac{C(x)}{opt}$ where $C(x) = \max_i \ell_i(x)$ and NE is the set of pure Nash equilibria. The SPoA is defined analogously w.r.t. the set SE of strong Nash equilibria: SPoA $= \max_{x \in \mathsf{SE}} \frac{C(x)}{opt}$.

While in the setting of *coordination mechanism*, the cost of each job is measured differently with the above load balancing game, because the jobs on the same machine are processed in a certain order (depending on the coordination mechanism) one by one, and so the cost is the completion time of each job. A coordination mechanism is a set of *local policies*, one for each machine. A local policy for machine j takes as input a set of jobs on machine j along with their information (e.g., size and favorite machine) and outputs an ordering in which they will be processed. Since the policy runs locally

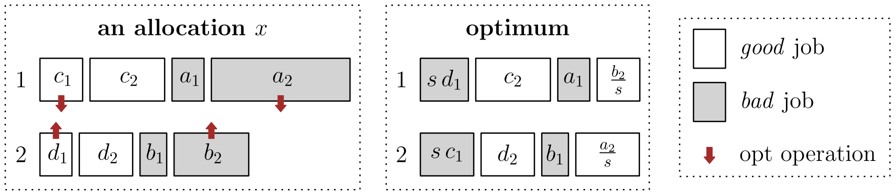


Fig. 2 Comparing SE with the optimum

at each machine, it does not have access to information regarding the global state of the system (e.g., the information of other jobs and machines). Given a certain coordination mechanism, the selfish jobs may induce some equilibria. The PoA of a coordination mechanism is the worst-case ratio between the social cost (i.e., makespan) of a NE and the optimum, given the coordination mechanism. We are interested in designing some coordination mechanism with small PoA, and at the same time, the coordination mechanism should be that the induced game always has pure Nash equilibria for any input. Furthermore, these equilibria should be easy to find.

2.2 First step of the analysis (reducing to eight groups of jobs)

To bound the (S-)PoA we have to compare the *worst case schedule* with the optimum. Given any allocation x, we consider the subset of jobs that needs to be reallocated in any allocation in order to obtain the optimum. It turns out that there are eight such subsets of jobs, and essentially (most of) the analyses are reduced to the cases of eight jobs only. All the following analyses in this paper are based on this grouping method.

We say that a job is *good*, for an allocation under consideration, if it is allocated to its *favorite* machine. Otherwise the job is *bad*. Consider an allocation x and the optimum, respectively. As shown in Fig. 2, let $\{a_1, a_2, c_1, c_2\}$ be 4 sets of jobs that are processed on machine 1, where $\{c_1, c_2\}$ are sets of good jobs and $\{a_1, a_2\}$ are sets of bad jobs. Similarly, let $\{b_1, b_2, d_1, d_2\}$ be 4 sets of jobs that are processed on machine 2, where $\{d_1, d_2\}$ are good jobs and $\{b_1, b_2\}$ are bad jobs. Note that each set also represent the total processing time of the jobs therein, e.g. a_1 is also the value of the total processing time of the jobs in a_1. Therefore, the loads of machine 1 and machine 2 in allocation x are

$$\ell_1 = a_1 + a_2 + c_1 + c_2 \quad \text{and} \quad \ell_2 = b_1 + b_2 + d_1 + d_2,$$

respectively, where $a_1 + a_2$ and $b_1 + b_2$ are load of bad jobs and $c_1 + c_2$ and $d_1 + d_2$ are load of good jobs. Without loss of generality suppose $\ell_1 \geq \ell_2$, that is

$$C(x) = \max(\ell_1, \ell_2) = \ell_1 \ .$$

In the optimum, some of the jobs will be processed by different machine as in allocation x. Suppose the jobs associated with a_2, b_2, c_1 and d_1 are the difference. Thus, the load of the two machines in the optimum are

$$\ell_1^* = a_1 + b_2/s + c_2 + s \cdot d_1 \quad \text{and} \quad \ell_2^* = a_2/s + b_1 + s \cdot c_1 + d_2.$$

Notice that each set is possibly empty and hence the corresponding value is 0.

Remark 1 The setting can be also applied to analyze PoA and SPoA for *two related machines* (Epstein 2010), which corresponds to two special cases $a_1, a_2, d_1, d_2 = 0$ and $b_1, b_2, c_1, c_2 = 0$ in our paper.

3 (Strong) price of anarchy for the makespan policy

We first focus on the SPoA for the Makespan policy, and then use the similar method to analyze the bound for PoA. To obtain tight bounds on SPoA, we exploit the conditions that possible reshuffling of these eight subsets must guarantee in order to be a SE.

3.1 Conditions for SE

Without loss of generality we suppose $opt = 1$, i.e.,

$$\ell_1^* \leq 1, \tag{1}$$
$$\ell_2^* \leq 1. \tag{2}$$

Since allocation x is SE, we have that the minimum load in allocation x must be at most opt,

$$\ell_2 \leq 1\ , \tag{3}$$

because otherwise we could swap some of the jobs to obtain opt and in this will improve the cost of all the jobs.

Next, we shall provide several necessary conditions for allocation x to be a SE. Though these conditions are only a subset of those that SE must satisfy, they will lead to tight bounds on the SPoA.

1. No job in machine 1 will go to machine 2:

$$\ell_1 \leq \ell_2 + a_1/s, \quad \text{for } a_1 > 0; \tag{4}$$
$$\ell_1 \leq \ell_2 + a_2/s, \quad \text{for } a_2 > 0; \tag{5}$$
$$\ell_1 \leq \ell_2 + s \cdot c_1, \quad \text{for } c_1 > 0; \tag{6}$$
$$\ell_1 \leq \ell_2 + s \cdot c_2, \quad \text{for } c_2 > 0. \tag{7}$$

2. No a_2 - b_2 swap:

$$\ell_1 \leq \ell_2 - b_2 + a_2/s, \tag{8}$$
$$\text{or} \quad \ell_2 \leq \ell_1 - a_2 + b_2/s. \tag{9}$$

3. No a_2 - d_1 swap:

$$\ell_1 \leq \ell_2 - d_1 + a_2/s, \tag{10}$$

Fig. 3 Subcases corresponding to Lemmas 2–6

$$\begin{cases} a_1 > 0 \quad \text{(Lemma 2)} \\ a_1 = 0 \quad \begin{cases} a_2 = 0 \quad \text{(Lemma 3)} \\ a_2 > 0 \quad \begin{cases} c_1 > 0 \quad \text{(Lemma 4)} \\ c_1 = 0 \quad \begin{cases} c_2 = 0 \quad \text{(Lemma 5)} \\ c_2 > 0 \quad \text{(Lemma 6)} \end{cases} \end{cases} \end{cases} \end{cases}$$

$$\text{or} \quad \ell_2 \le \ell_1 - a_2 + s \cdot d_1. \tag{11}$$

4. No a_2 - $\{b_1, d_2\}$ swap:

$$\ell_1 \le \ell_2 - b_1 - d_2 + a_2/s, \tag{12}$$

$$\text{or} \quad \ell_2 \le \ell_1 - a_2 + b_1/s + s \cdot d_2. \tag{13}$$

5. No $\{a_2, c_2\}$ - $\{b_1, b_2, d_1, d_2\}$ swap:

$$\ell_1 \le a_2/s + c_2 \cdot s, \tag{14}$$

$$\text{or} \quad \ell_2 \le a_1 + c_1 + (b_1 + b_2)/s + s(d_1 + d_2). \tag{15}$$

6. No $\{a_1, a_2, c_1, c_2\}$ - $\{b_1, b_2, d_1, d_2\}$ swap:

$$\ell_1 \le (a_1 + a_2)/s + s(c_1 + c_2), \tag{16}$$

$$\text{or} \quad \ell_2 \le (b_1 + b_2)/s + s(d_1 + d_2). \tag{17}$$

3.2 Tight bounds for SPoA

Theorem 1 SPoA $= \hat{\ell}_1$, *where (see the blue line in Fig. 1)*

$$\hat{\ell}_1 = \begin{cases} \frac{s^3+s^2+s+1}{s^3+2}, & 1 \le s \le s_1 \approx 1.325 \\ \frac{s^2+2s+1}{2s+1}, & s_1 \le s \le s_2 = \frac{1+\sqrt{5}}{2} \approx 1.618 \\ \frac{s+1}{s}, & s_2 \le s \le s_3 \approx 1.755 \\ \frac{s^3-s^2+2s-1}{s^3-s^2+s-1}, & s_3 \le s \le s_4 \approx 1.907 \\ \frac{s+1}{2}, & s_4 \le s \le s_5 = 2 \\ \frac{s^2-s+1}{s^2-s}, & s_5 \le s \le s_6 \approx 2.154 \\ \frac{s^2}{2s-1}, & s_6 \le s \le s_7 \approx 2.247 \\ \frac{s+1}{s}, & s_7 \le s. \end{cases}$$

We break the proof of Theorem 1 into several subcases (Lemmas 2–6), and prove the upper bound in each of them. The lemmas are organized as shown in Fig. 3 depending on the value of a_1, a_2, c_1, c_2. Finally, we show that the bounds of these lemmas are tight (Lemma 7).

Lemma 2 *If* $a_1 > 0$, *then*

$$\ell_1 \le \begin{cases} \frac{2(s+1)}{s+2}, & 1 \le s \le \sqrt{2} \\ \frac{s(s^2+s-1)}{s^3-s+1}, & \sqrt{2} \le s \le \frac{1+\sqrt{5}}{2} \\ \frac{2s}{2s-1}, & \frac{1+\sqrt{5}}{2} \le s \le 1.905 \\ \frac{s^4+s^3-1}{s^4+s-1}, & s \ge 1.905. \end{cases}$$

Lemma 3 *If* $a_1 = a_2 = 0$, *then*

$$\ell_1 \le \begin{cases} \frac{2(s+1)}{s+2}, & 1 \le s \le \sqrt{2} \\ \frac{s+2}{s+1}, & s \ge \sqrt{2}. \end{cases}$$

Lemma 4 *If* $a_1 = 0$, $a_2, c_1 > 0$, *then*

$$\ell_1 \le \begin{cases} \frac{s^3+s^2+2s+2}{s^2+2s+2}, & 1 \le s \le \sqrt{2} \\ \frac{2s^2+s-2}{s^2+s-1}, & \sqrt{2} \le s \le \frac{1+\sqrt{5}}{2} \\ \frac{s^2-s+2}{s^2-s+1}, & \frac{1+\sqrt{5}}{2} \le s \le 2 \\ \frac{s^2}{s^2-s+1}, & s \ge 2. \end{cases}$$

Lemma 5 *If* $a_1 = c_1 = c_2 = 0$ *and* $a_2 > 0$, *then*

$$\ell_1 \le \begin{cases} s, & 1 \le s \le \frac{1+\sqrt{5}}{2} \\ \frac{s(s^2+s-1)}{s^3-1}, & \frac{1+\sqrt{5}}{2} \le s \le 1.861 \\ \frac{s^2}{2s^2-3s+1}, & 1.861 \le s \le 2 \\ \frac{s^2}{2s-1}, & 2 \le s \le 2.247 \\ \frac{s+1}{s}, & s \ge 2.247. \end{cases}$$

Lemma 6 *If* $a_1 = c_1 = 0$ *and* $a_2, c_2 > 0$, *then*

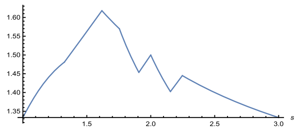
$$\ell_1 \le \begin{cases} \frac{s^3+s^2+s+1}{s^3+2}, & 1 \le s \le 1.325 \\ \frac{s^2+2s+1}{2s+1}, & 1.325 \le s \le \frac{1+\sqrt{5}}{2} \\ \frac{s+1}{s}, & \frac{1+\sqrt{5}}{2} \le s \le 1.755 \\ \frac{s^3-s^2+2s-1}{s^3-s^2+s-1}, & 1.755 \le s \le 1.907 \\ \frac{s+1}{2}, & 1.907 \le s \le 2 \\ \frac{s^2-s+1}{s^2-s}, & 2 \le s \le 2.277 \\ \frac{s^3+s^2+s+1}{s^3+s+1}, & s \ge 2.277. \end{cases}$$

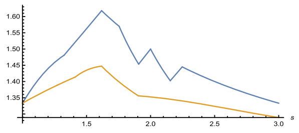
Lemma 7 *The lower bound of* SPoA $\ge \hat{\ell}_1$ *is given by the instances in Table* 2.

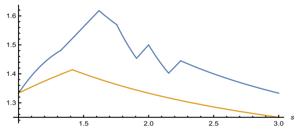
Table 2 Lower bound for SPoA

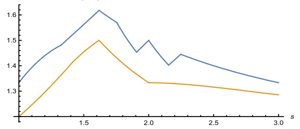
	s	a_2	b_2	c_2	d_1	d_2	LB ($=\ell_1$)	ℓ_2
LB1	$[0, s_1]$	$\frac{s^3+s^2}{s^3+2}$	$\frac{s}{s^3+2}$	$\frac{s+1}{s^3+2}$	$\frac{s^2-1}{s^3+2}$	$\frac{s^3-s^2-s+2}{s^3+2}$	$\frac{s^3+s^2+s+1}{s^3+2}$	$\frac{s^3+1}{s^3+2}$
LB2	$[s_1, s_2]$	$\frac{s^2+s}{2s+1}$	$\frac{s^2}{2s+1}$	$\frac{s+1}{2s+1}$	0	$\frac{s}{2s+1}$	$\frac{s^2+2s+1}{2s+1}$	$\frac{s^2+s}{2s+1}$
LB3	$[s_2, s_3]$	1	$s-1$	$\frac{1}{s}$	0	$2-s$	$\frac{s+1}{s}$	1
LB4	$[s_3, s_4]$	$\frac{s^2/(s-1)}{s^2+1}$	$\frac{s^2}{s^2+1}$	$\frac{s^2-s+1}{s^2+1}$	0	$\frac{1}{s^2+1}$	$\frac{s^3-s^2+2s-1}{s^3-s^2+s-1}$	1
LB5	$[s_4, s_5]$	$\frac{s}{2}$	$\frac{s}{2}$	$\frac{1}{2}$	0	$\frac{2-s}{2}$	$\frac{s+1}{2}$	1
LB6	$[s_5, s_6]$	$\frac{1}{s-1}$	1	$\frac{s-1}{s}$	0	0	$\frac{s^2-s+1}{s^2-s}$	1
LB7	$[s_6, s_7]$	$\frac{s^2}{2s-1}$	0	0	$\frac{(s-1)^2}{2s-1}$	$\frac{s-1}{2s-1}$	$\frac{s^2}{2s-1}$	$\frac{s^2-s}{2s-1}$
LB8	$[s_7, \infty]$	$\frac{s+1}{s}$	0	0	$\frac{1}{s}$	$\frac{s^2-s-1}{s^2}$	$\frac{s+1}{s}$	$\frac{s^2-1}{s^2}$

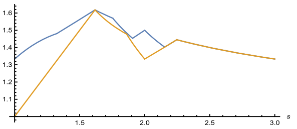
Note that $a_1 = b_1 = c_1 = 0$, and a_2, b_2, c_2, d_1, d_2 each represent a single job here


(a) Theorem 1.


(b) Lemma 2.


(c) Lemma 3.


(d) Lemma 4.


(e) Lemma 5.

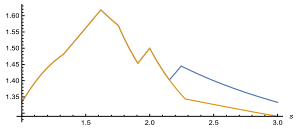

(f) Lemma 6.

Fig. 4 Proof of Theorem 1 (Color figure online)

Figure 4 illustrates the relation between the general bound (blue line) of Theorem 1 and the bounds (orange line) proved in each of the these lemmas. It is easy to see that the bound of Theorem 1 is just the union of the bounds of Lemmas 2–6. Thus we only need to prove the lemmas.

The proofs of the Lemmas 2–6 are based on Table 3, which lists all the subcases that need to be considered. Each subcase is corresponding to a row of the table which contains the constraints used (column 4) to obtain the bound of the subcase (column 6). Note that for the chosen constraints, we also specify some *weight* (column 5) which essentially says how these constraints are combined together to get the desired bound. We explain it with the following example.

Table 3 Subcases to prove Lemmas 2–6 and Lemmas 10–11

Lemma.subcases	s	a_1, a_2, c_1, c_2	Constrains needed	Weight coefficient	Bounds
2.1	$[1, \infty]$	$1,\ 0,\ 0,\ *$	(1)	$\{1\}$	1
2.2	$[1, \infty]$	$1,\ 0,\ 1,\ *$	(1, (2), (4), (6)	$\frac{\{s^3+s;s^2+1;s^2;1\}}{s^3+s^2+1}$	$\frac{s^3+s^2+s+1}{s^3+s^2+1}$
2.3-a	$[1, \sqrt{2}]$	$1,\ 1,\ 0,\ *$	(1, (2), (4), (5)	$\{2s;\ 2;\ s^2;\ 2-s^2\} \cdot \frac{1}{s+2}$	$\frac{2(s+1)}{s+2}$
2.3-b	$[\sqrt{2}, \infty]$	$1,\ 1,\ 0,\ *$	(1, (2), (3), (4)	$\frac{\{s^2;s;s(s^2-2);s(s^2-1)\}}{s^3-s+1}$	$\frac{s(s^2+s-1)}{s^3-s+1}$
2.3-c	$[1, \infty]$	$1,\ 1,\ 0,\ *$	(2, (4), (5)	$\{2s;\ s;\ s\} \cdot \frac{1}{2s-1}$	$\frac{2s}{2s-1}$
2.4-a	$[1, 1.272]$	$1,\ 1,\ 1,\ *$	(1, (2), (4), (5), (6)	$\frac{\{2s^3+s;2s^2+1;s^4;-s^4+s^2+1;s^2\}}{s^3+2s^2+s+1}$	$\frac{2s^3+2s^2+s+1}{s^3+2s^2+s+1}$
2.4-b	$[1.0][1.272, \infty]$	$1,\ 1,\ 1,\ *$	(1, (2), (3), (4), (6)	$\frac{\{s^3;s^2;s^4-s^2-1;s^2(s^2-1);s^2-1\}}{s^4+s-1}$	$\frac{s^4+s^3-1}{s^4+s-1}$
3.1	$[1, \infty]$	$0,\ 0,\ 0,\ 1$	(1)	$\{1\}$	1
3.2	$[1, \infty]$	$0,\ 0,\ 1,\ 0$	(2)	$\{1/s\}$	$1/s$
3.3-a	$[1, \infty]$	$0,\ 0,\ 1,\ 1$	(1, (2), (6), (7)	$\{2s;\ 2;\ 1;\ 1\} \cdot \frac{1}{s+2}$	$\frac{2(s+1)}{s+2}$
3.3-b	$[1, \infty]$	$0,\ 0,\ 1,\ 1$	(1, (2), (6)	$\{s;\ 2;\ 1\} \cdot \frac{1}{s+1}$	$\frac{s+2}{s+1}$
4.1-a	$[1, \infty]$	$0,\ 1,\ 1,\ 0$	(2, (3), (6)	$\frac{\{s^2;s^2-1;s^2-1\}}{s^2+s-1}$	$\frac{2s^2-1}{s^2+s-1}$
4.1-b.(8)	$[1.272, \infty]$	$0,\ 1,\ 1,\ 0$	(1, (2), (3), (6), (8)	$\frac{\{s^3;s^4;s^4-s^2-1;s^4-1;s^4-s^2\}}{2s^4-s^2+s-1}$	$\frac{2s^4+s^3-s^2-1}{2s^4-s^2+s-1}$
4.1-b.(9)	$[1.272, \infty]$	$0,\ 1,\ 1,\ 0$	(2, (5), (9)	$\frac{\{s^2-s+1;(s-1)^2(s+1);s(s^2-1)\}}{s^3-2s^2+s+1}$	$\frac{s^2-s+1}{s^3-2s^2+s+1}$
4.2-a.(14)	$[1, \infty]$	$0,\ 1,\ 1,\ 1$	(3, (5), (6), (14)	$\{s^2;\ s^2-1;\ 1;\ 1\} \cdot \frac{1}{s^2-s+1}$	$\frac{s^2}{s^2-s+1}$
4.2-a.(15)	$[1, \infty]$	$0,\ 1,\ 1,\ 1$	(3, (5), (7), (15)	$\frac{\{s^2-s+2;s^2;1;s\}}{s^2-s+1}$	$\frac{s^2-s+2}{s^2-s+1}$
4.2-b	$[1, \sqrt{2}]$	$0,\ 1,\ 1,\ 1$	(1, (2), (5), (6), (7)	$\frac{\{s(s^2+2);s^2+2;2-s^2;s^2;s^2\}}{s^2+2s+2}$	$\frac{s^3+s^2+2s+2}{s^2+2s+2}$
4.2-c	$[\sqrt{2}, \infty]$	$0,\ 1,\ 1,\ 1$	(1, (2), (3), (6)	$\frac{\{s;s^2;s^2-2;s^2-1\}}{s^2+s-1}$	$\frac{2s^2+s-2}{s^2+s-1}$
5-a	$[1, \frac{1+\sqrt{5}}{2}]$	$0,\ 1,\ 0,\ 0$	(2)	$\{s\}$	s

Table 3 continued

Lemma.subcases	s	a_1, a_2, c_1, c_2	Constrains needed	Weight coefficient	Bounds
5-b.(8)-a	$[1, \infty]$	0, 1, 0, 0	(1, (2), (5), (8)	$\{s;\, s^2;\, 1;\, s^2-1\} \cdot \frac{1}{s^2}$	$1+\frac{1}{s}$
5-b.(8)-b.(12)-a	$[1, \infty]$	0, 1, 0, 0	(1, (2), (8), (12)	$\frac{\{s^2; s(s^2-1); s(s^2-1); s\}}{s^3-1}$	$\frac{s(s^2+s-1)}{s^3-1}$
5-b.(8)-b.(12)-b.(10)	$[1, \infty]$	0, 1, 0, 0	(2, (3), (8), (10)	$\left\{\frac{s}{2s-1};\, \frac{s}{2s-1};\, \frac{s}{2s-1};\, \frac{s}{2s-1}\right\}$	$\frac{2s}{2s-1}$
5-b.(8)-b.(12)-b.(11)	$[1, \infty]$	0, 1, 0, 0	(1, (8), (11), (12)	$\frac{\{s^2; s^2-s; s^2-s; s^2\}}{2s^2-3s+1}$	$\frac{s^2}{2s^2-3s+1}$
5-b.(8)-b.(13)	$[1, \infty]$	0, 1, 0, 0	(2, (5), (13)	$\{s^2;\, s;\, s\} \cdot \frac{1}{2s-1}$	$\frac{s^2}{2s-1}$
5-b.(9)	$[1, \infty]$	0, 1, 0, 0	–	–	–
6.(16)-a	$[1, \infty]$	0, 1, 0, 1	(1, (2), (5), (16)	$\frac{\{s(s+1); s+1; s+1; s^2/(s-1)\}}{2s+1}$	$\frac{(s+1)^2}{2s+1}$
6.(16)-b	$[1, \infty]$	0, 1, 0, 1	(3, (5), (16)	$\left\{\frac{s+1}{s};\, \frac{s+1}{s};\, \frac{1}{(s-1)s}\right\}$	$1+\frac{1}{s}$
6.(16)-c.(8)	$[1, \infty]$	0, 1, 0, 1	(1, (2), (5), (7), (8)	$\frac{\{s^2+1; s^3+s; 1/s; s; s^3-1/s\}}{s^3+s+1}$	$\frac{s^3+s^2+s+1}{s^3+s+1}$
6.(16)-c.(9)-a	$[1, \infty]$	0, 1, 0, 1	(3, (5), (9)	$\left\{\frac{s^2-s+1}{(s-1)s};\, \frac{s}{s-1};\, \frac{1}{s-1}\right\}$	$\frac{s^2-s+1}{(s-1)s}$
6.(16)-c.(9)-b.(12)	$[1, \infty]$	0, 1, 0, 1	(1, (12), (16)	$\left\{\frac{s+1}{2};\, \frac{s+1}{2s};\, \frac{s^2+1}{2(s-1)s}\right\}$	$\frac{s+1}{2}$
6.(16)-c.(9)-b.(13)	$[1, \infty]$	0, 1, 0, 1	(1, (3), (5), (9), (13)	$\frac{\{(s-1)(s^2+1); s; s(s^2+1); s^3+s-1; 1\}}{s^3-s^2+s-1}$	$\frac{s^3-s^2+2s-1}{s^3-s^2+s-1}$
6.(17)-a.(10)	$[1, \infty]$	0, 1, 0, 1	(1, (2), (17), (10)	$\frac{\{s; s^2+s+1; s^2/(s-1); s+1\}}{2s+1}$	$\frac{(s+1)^2}{2s+1}$
6.(17)-a.(11)	$[1, \infty]$	0, 1, 0, 1	(1, (2), (5), (7), (11)	$\frac{\{s^3+s; s^2+1; s^3+s^2-s; s+1-\frac{1}{s}; s^3-\frac{1}{s}\}}{s^3+2}$	$\frac{s^3+s^2+s+1}{s^3+2}$
6.(17)-b.(8)	$[1, \infty]$	0, 1, 0, 1	(1, (2), (5), (7), (8)	$\frac{\{s^2+1; s^3+s; 1/s; s; s^3-1/s\}}{s^3+s+1}$	$\frac{s^3+s^2+s+1}{s^3+s+1}$
6.(17)-b.(9)	$[1, \infty]$	0, 1, 0, 1	(3, (5), (9), (17)	$\frac{\{s^2;\, s(s+1);\, s+1;\, 1/(s-1)\}}{s^2-1}$	$\frac{s^2}{s^2-1}$

Table 3 continued

Lemma.subcases	s	a_1, a_2, c_1, c_2	Constrains needed	Weight coefficient	Bounds
10.1-a	$[1, \sqrt{2}]$	$1,\ 1,\ *,\ 0$	(1, (2), (4), (5)	$\{2s; 2; s^2; 2 - s^2\} \cdot \frac{1}{s+2}$	$\frac{2(s+1)}{s+2}$
10.1-b	$[\sqrt{2}, \infty]$	$1,\ 1,\ *,\ 0$	(1, (2), (4)	$\{s^2; s(s^2 - 1); s\} \cdot \frac{1}{s^2+s-1}$	s
10.2	$[1, \infty]$	$0,\ 1,\ 1,\ 0$	(1, (2), (6)	$\frac{\{s(s^2-1); s^2; s^2-1\}}{s^2+s-1}$	s
11	$[1, \infty]$	$*,\ 1,\ *,\ 1$	(1, (2), (5), (7)	$\frac{\{s^3+s; s^2+1; 1; s^2\}}{s^2+s+1}$	$\frac{s^3+s^2+s+1}{s^2+s+1}$

"$*$" means *either* 0 *or* 1 in the third column

An illustrative example (weighted combination of constraints). Consider the case 2-2 in Table 3 (second row). In the third column, the four numbers show whether the four variable a_1, a_2, c_1, c_2 are zero or non-zero, where "1" represents non-zero and "$*$" represents non-negative. The last column is the bound we obtain for ℓ_1 and thus for the SPoA. Specifically, in this case we want to prove the following:

Claim 1 *If* $a_1 > 0$, $a_2 = 0$, $c_1 > 0$ *and* $s \geq 1$, *then* $\ell_1 \leq \frac{s^3+s^2+s+1}{s^3+s^2+1}$.

Proof First summing all the constraints with the corresponding weights given in columns 4 and 5 of Table 3 (row 2-2):

$$\frac{s^3+s}{s^3+s^2+1} \cdot (1) + \frac{s^2+1}{s^3+s^2+1} \cdot (2) + \frac{s^2}{s^3+s^2+1} \cdot (4) + \frac{1}{s^3+s^2+1} \cdot (6).$$

This can be simplified as

$$a_1 + \frac{s^3+s^2+s+1}{s^4+s^3+s} a_2 + c_1 + \frac{s^3+s^2+s+1}{s^3+s^2+1} c_2 + \frac{s^4-1}{s^3+s^2+1} d_1 \leq \frac{s^3+s^2+s+1}{s^3+s^2+1}. \tag{18}$$

Note that all the weights (column 5) are positive when s is within the given interval (column 2), so that the direction of the inequalities (column 4) remains.

According to columns 2 and 3, we have $a_2 = 0$, $\frac{s^3+s^2+s+1}{s^3+s^2+1} \geq 1$ and $\frac{s^4-1}{s^3+s^2+1} \geq 0$. Along with (18), it implies that

$$\ell_1 = a_1 + a_2 + c_1 + c_2 \leq \frac{s^3+s^2+s+1}{s^3+s^2+1}.$$

We therefore get the bound in column 6. □

Since the above illustrative example has explained how the bounds are generated, then we will show the relationship of these bounds with the lemmas. The index (column 1) of each row in Table 3 encodes the relationship between those bounds (column 6) and how these bounds should be combined to obtain the corresponding lemma. Specifically, cases separated by "." are subcases that should take *maximum* of them since we aim to measure the worst performance, while cases separated by "-" are a single case bounded by several different combinations of constrains that should take *minimum* of them since these constrains should hold at the same time. For instance, we consider three subcases in Lemma 3, since $a_1 = a_2 = 0$: Case 3.1 ($c_1 = 0, c_2 > 0$), Case 3.2 ($c_1 > 0, c_2 = 0$) and Case 3.3 ($c_1 > 0, c_2 > 0$), which refer to rows 3.1 to 3.3-b in Table 3, respectively. The bound for Case 3.3 is the *minimum* of the bounds of 3.3-a and 3.3-b because of "-". Finally, due to cases 3.1, 3.2 and 3.3 are separated by ".", we shall take the *maximum* of the bounds of the three cases which gives the bound for Lemma 3, i.e., $\max\left\{1, \frac{1}{s}, \min\{\frac{2(s+1)}{s+2}, \frac{s+2}{s+1}\}\right\} = \min\{\frac{2(s+1)}{s+2}, \frac{s+2}{s+1}\}$ (orange line of Fig. 4c).

The proofs of Lemmas 2–7 are given in full detail in "Appendices A.1 and A.2".

3.3 Price of anarchy

Next we give *exact* bounds on the PoA, which shows that the case in Example 1 is not even the worst case.

Theorem 8 $\text{PoA} = \frac{s^3+s^2+s+1}{s^2+s+1}$ *(see the orange line in* Fig. 1*).*

Proof Suppose the smallest jobs in a_1, a_2, c_1, c_2 are $a_1{}', a_2{}', c_1{}', c_2{}'$ respectively. To guarantee a NE, it must hold that no single job in machine 1 can improve by moving to machine 2, so that (4), (5), (6) and (7) are also true for NE since $a_1{}' \le a_1$, $a_2{}' \le a_2$, $c_1{}' \le c_1$ and $c_2{}' \le c_2$. Similar to the analysis of the SPoA, we assume without loss of generality that $opt = 1$, and thus (1) and (2) hold. We use these six constraints to prove Theorem 8.

It is easy to verify that if at most one of a_1, a_2, c_1, c_2 is nonzero, then $\ell_1 \le s$, thus here we only discuss the cases where at least two of them are nonzero. Similar to the proof of SPoA the proofs of these lemmas are based on last four rows of Table 3.

Lemma 9 *If* $a_2 = 0$*, then* $\ell_1 \le \frac{s^3+s^2+s+1}{s^2+s+1}$.

Lemma 10 *If* $a_2 > 0$ *and* $c_2 = 0$*, then* $\ell_1 \le \frac{s^3+s^2+s+1}{s^2+s+1}$.

Lemma 11 *If* $a_2 > 0$ *and* $c_2 > 0$*, then* $\ell_1 \le \frac{s^3+s^2+s+1}{s^2+s+1}$.

Lemma 12 *The lower bound of* PoA *is achieved by the following case,*

$$a_2 = \frac{s^3+s^2}{s^2+s+1}, \quad b_1 = \frac{1}{s^2+s+1}, \quad b_2 = \frac{s^3}{s^2+s+1}, \quad c_2 = \frac{s+1}{s^2+s+1},$$

and $a_1 = c_1 = d_1 = d_2 = 0$.

Lemmas 9–12 (proofs see Appendix A.3) complete the proof of Theorem 8. □

These bounds express the dependency on the parameter s and suggest a natural comparison with the case of two related machines (with the same s).

4 Coordination mechanism design

In this section, we design a new coordination mechanism for the problem. Since our problem is a generalization of the two related machines, the PoA of the SPT policy for our problem is no less than 1.618 (the PoA of SPT for the two related machines). For the LPT policy, Example 1 already shows that the PoA is unbounded. Thus, we propose a new policy, called the FF-LPT, which achieves a better PoA than any known policies.

4.1 The FF-LPT policy

The FF-LPT (Favorite First-Longest Processing Time) policy is defined as follows: each machine first processes the good jobs in LPT rule (decreasing order in processing time), then processes the bad jobs in LPT rule.

We adopt the same setting in Sect. 2.2 (Fig. 2). Note that Fig. 2 only shows how the jobs are partitioned into several subsets, but does not reflect the actual process order of the jobs. Without loss of generality, we assume that $opt \leq 1$, i.e., $\ell_1^* \leq 1$ and $\ell_2^* \leq 1$.

Observation 1 $(a_1 + a_2)(b_1 + b_2) = 0$.

Proof by Contradiction. Suppose $(a_1 + a_2)(b_1 + b_2) \neq 0$. As a_1, a_2, b_1 and b_2 are nonnegative, we can get that $a_1 + a_2 > 0$ and $b_1 + b_2 > 0$. Since $a_1 + a_2 > 0$, we have $d_1 + d_2 > c_1 + c_2$; Otherwise, any job in a_1 or a_2 would benefit by moving to machine 2, because any job in a_1 or a_2 is a bad job and has a smaller starting time and a smaller processing time on machine 2 (i.e., smaller completion time). Similarly, since $b_1 + b_2 > 0$, we have $c_1 + c_2 > d_1 + d_2$, contradicting the above inequation $d_1 + d_2 > c_1 + c_2$. Therefore, it holds that $(a_1 + a_2)(b_1 + b_2) = 0$. □

According to Observation 1, we consider the following three cases:

- Case 1: $a_1 + a_2 = b_1 + b_2 = 0$;
- Case 2: $a_1 + a_2 = 0$ and $b_1 + b_2 > 0$;
- Case 3: $a_1 + a_2 > 0$ and $b_1 + b_2 = 0$.

Lemma 13 *For Case 1 ($a_1 + a_2 = b_1 + b_2 = 0$), $\ell_1 \leq \frac{4}{3}$.*

Proof As $a_1 + a_2 = b_1 + b_2 = 0$, we know that $\ell_1 = c_1 + c_2$, $\ell_2 = d_1 + d_2$, $\ell_1^* = c_2 + s \cdot d_1$ and $\ell_2^* = s \cdot c_1 + d_2$. We first argue that if $c_1 = 0$ (or $c_2 = 0$), it holds that $\ell_1 \leq 1$. When $c_1 = 0$, we have $\ell_1 = c_2 \leq c_2 + s \cdot d_1 = \ell_1^* \leq 1$. When $c_2 = 0$, we have $\ell_1 = c_1 \leq s \cdot c_1 + d_2 = \ell_2^* \leq 1$. Therefore, we consider the case $c_1, c_2 > 0$ (none of c_1 and c_2 is empty set) in the following.

Let $c_{\min}$ be the job with the smallest processing time among the job sets c_1 and c_2. Obviously, $c_{\min} \leq c_1$ and $c_{\min} \leq c_2$. According to the FF-LPT policy, we know that job $c_{\min}$ must be the last processed job on machine 1 (i.e., the cost of the job is $\ell_1 = c_1 + c_2$), and it would be the last processed job on machine 2 if it chose machine 2. To guarantee the allocation is a NE, it holds that $\ell_1 \leq \ell_2 + s \cdot c_{\min}$ (job $c_{\min}$ is a bad job for machine 2). To prove that $\ell_1 \leq \frac{4}{3}$, we only need to solve the following maximization problem:

$$
\begin{aligned}
\max \quad & \ell_1 = c_1 + c_2 & & \\
s.t. \quad & c_2 + s \cdot d_1 \leq 1 & (\ell_1^* \leq 1) \quad & (19)\\
& s \cdot c_1 + d_2 \leq 1 & (\ell_2^* \leq 1) \quad & (20)\\
& c_1 + c_2 \leq d_1 + d_2 + s \cdot c_{\min} & (\ell_1 \leq \ell_2 + s \cdot c_{\min}) \quad & (21)\\
& c_{\min} \leq c_1 & & (22)\\
& c_{\min} \leq c_2 & & \\
& s \geq 1. & & (23)
\end{aligned}
$$

Combine these constraints (19)–(23) with proper weights in order to eliminate d_1, d_2 and $c_{\min}$ and equal the coefficients of c_1 and c_2, and we obtain that

$$
\begin{aligned}
&\frac{1}{s}\cdot(19)+(20)+(21)+\left(s-\frac{1}{2s}\right)\cdot(22)+\frac{1}{2s}\cdot(23)\\
&\quad\Rightarrow\quad\left(1+\frac{1}{2s}\right)(c_1+c_2)\le 1+\frac{1}{s}\\
&\quad\Rightarrow\quad \ell_1=c_1+c_2\le\frac{1+\frac{1}{s}}{1+\frac{1}{2s}}\le\frac{4}{3}.
\end{aligned}
$$

Lemma 14 *For Case 2 ($a_1+a_2=0$ and $b_1+b_2>0$), $\ell_1\le\frac{4}{3}$.*

Proof As $a_1+a_2=0$, we know that $\ell_1=c_1+c_2$, $\ell_2=d_1+d_2+b_1+b_2$, $\ell_1^*=b_2/s+c_2+s\cdot d_1$ and $\ell_2^*=b_1+s\cdot c_1+d_2$. Let $c_{\min}$ be the job with the smallest processing time among the job sets c_1 and c_2, i.e., job $c_{\min}$ is the last processed job on machine 1 with completion time ℓ_1. Since job $c_{\min}$ is a bad job for machine 2, if it moved to machine 2, it would be processed after all the good jobs (i.e., d_1+d_2), and also after the jobs in b_1+b_2 whose processing time is greater than $s\cdot c_{\min}$. Two subcases arise depending on the position of job $c_{\min}$ if it moves to machine 2:

(1) *The processing time of job $c_{\min}$ on machine 2 is greater than any job in b_1+b_2, i,e., job $c_{\min}$ will be processed before all jobs in b_1+b_2.*

To guarantee that job $c_{\min}$ has no incentive to move to machine 2, it holds that $\ell_1\le d_1+d_2+s\cdot c_{\min}$. Because $\ell_1^*,\ell_2^*\le 1$ and $b_1,b_2\ge 0$, we have $c_2+s\cdot d_1\le 1$ and $s\cdot c_1+d_2\le 1$. Therefore, as the constraints (19)–(23) also hold here, we can obtain the same bound of $\ell_1\le\frac{4}{3}$.

(2) *There is at least one job in b_1+b_2 that has processing time greater than $s\cdot c_{\min}$ so that it is processed before job $c_{\min}$.*

Suppose job $c_{\min}$ will be processed at time $d_1+d_2+b_s$ if job $c_{\min}$ moves to machine 2, where b_s denotes the total processing time of jobs in b_1+b_2 whose processing time is greater than $s\cdot c_{\min}$. Obviously, $s\cdot c_{\min}\le b_s\le b_1+b_2$.

To guarantee that job $c_{\min}$ has no incentive to move to machine 2, it holds that $\ell_1\le d_1+d_2+b_s+s\cdot c_{\min}\le d_1+d_2+2(b_1+b_2)$. Along with $\ell_1^*\le 1$ and $\ell_2^*\le 1$, we have

$$
\begin{aligned}
\max\quad & \ell_1=c_1+c_2\\
s.t.\quad & b_2/s+c_2+s\cdot d_1\le 1 && (24)\\
& b_1+s\cdot c_1+d_2\le 1 && (25)\\
& c_1+c_2\le d_1+d_2+2(b_1+b_2)\\
& s\ge 1. && (26)
\end{aligned}
$$

Combining these constraints (24)–(26) with proper weights, we get:

$$
\begin{aligned}
&2s\cdot(24)+2\cdot(25)+(26)\\
&\quad\Rightarrow\quad(2s+1)(c_1+c_2)+(2s^2-1)d_1+d_2\le 2s+2\\
&\quad\Rightarrow\quad \ell_1=c_1+c_2\le\frac{2s+2}{2s+1}\le\frac{4}{3}.
\end{aligned}
$$

Lemma 15 *For Case 3 ($a_1 + a_2 > 0$ and $b_1 + b_2 = 0$), $\ell_1 \leq \frac{4}{3}$.*

Proof As $b_1 + b_2 = 0$, we know that $\ell_1 = a_1 + a_2 + c_1 + c_2$, $\ell_2 = d_1 + d_2$, $\ell_1^* = a_1 + c_2 + s \cdot d_1$ and $\ell_2^* = a_2/s + s \cdot c_1 + d_2$. Let $a_{\min}$ be the job with the smallest processing time among the jobs in $a_1 + a_2$, i.e., job $a_{\min}$ is the last processed job on machine 1 with completion time ℓ_1. Since job $a_{\min}$ is a good job for machine 2, if it moved to machine 2, it would be processed after the jobs in $d_1 + d_2$ whose processing time is greater than $a_{\min}/s$ and before the ones whose processing time is less than $a_{\min}/s$. Two subcases arise depending on the position of job $a_{\min}$ if it moves to machine 2:

(1) *When job $a_{\min}$ moves to machine 2, it is the last processed job on machine 2, i.e., job $a_{\min}$ is processed after all jobs in $d_1 + d_2$.*

In this case, we know that $d_1 + d_2 > 0$; otherwise any job in $a_1 + c_2$ can benefit by moving to machine 2. The proof is further separated into several subcases:

(1-1) The case when $a_2 = 0$. Because job $a_{\min}$ will be the last job if it moves to machine 2, we have that $a_{\min}/s$ is smaller than the processing time of any job in $d_1 + d_2$. Thus it holds that $a_{\min}/s \leq d_1 + d_2$. To guarantee job $a_{\min}$ will not move to machine 2, we have $a_1 + c_1 + c_2 \leq d_1 + d_2 + a_{\min}/s \leq 2(d_1 + d_2)$. Along with $\ell_1^* \leq 1$ and $\ell_2^* \leq 1$, we have

$$\begin{aligned}
\max \quad & \ell_1 = a_1 + c_1 + c_2 \\
s.t. \quad & a_1 + c_2 + s \cdot d_1 \leq 1 && (27)\\
& s \cdot c_1 + d_2 \leq 1 && (28)\\
& a_1 + c_1 + c_2 \leq 2(d_1 + d_2) \\
& s \geq 1. && (29)
\end{aligned}$$

Combining these constraints (27)–(29) with proper weights, we get:

$$\begin{aligned}
& 2s \cdot (27) + 2 \cdot (28) + (29) \\
& \Rightarrow (2s + 1)(a_1 + c_1 + c_2) + (2s^2 - 2)d_1 \leq 2s + 2 \\
& \Rightarrow \ell_1 = a_1 + c_1 + c_2 \leq \frac{2s + 2}{2s + 1} \leq \frac{4}{3}.
\end{aligned}$$

(1-2) The case when $a_2 > 0$ and $a_1 > 0$. Due to $a_2 > 0$, $a_1 > 0$ and the definition of job $a_{\min}$, we have $a_{\min} \leq a_1$ and $a_{\min} \leq a_2$. Thus,

$$\begin{aligned}
\max \quad & \ell_1 = a_1 + a_2 + c_1 + c_2 \\
s.t. \quad & a_1 + c_2 + s \cdot d_1 \leq 1 && (30)\\
& a_2/s + s \cdot c_1 + d_2 \leq 1 && (31)
\end{aligned}$$

$$a_1 + a_2 + c_1 + c_2 \le d_1 + d_2 + a_{\min}/s \tag{32}$$
$$a_{\min} \le a_1 \tag{33}$$
$$a_{\min} \le a_2$$
$$s \ge 1. \tag{34}$$

Combining these constraints (30)–(34) with proper weights, we get:

$$\frac{1}{s} \cdot (35) + (31) + (32) + \frac{1}{2s} \cdot (33) + \frac{1}{2s} \cdot (34)$$
$$\Rightarrow \left(\frac{1}{2s} + 1\right)(a_1 + a_2 + c_1 + c_2) + \left(s - \frac{1}{2s}\right) c_1 + \frac{1}{2s} c_2 \le \frac{1}{s} + 1$$
$$\Rightarrow \ell_1 = a_1 + a_2 + c_1 + c_2 \le \frac{2s+2}{2s+1} \le \frac{4}{3}.$$

(1-3) The case when $a_2 > 0$ and $a_1 = 0$. We first consider the trivial case when $d_1 = 0$. We know that $\ell_2^* = a_2/s + s \cdot c_1 + d_2 \le 1$ and $a_{\min} \le a_2$. To guarantee job $a_{\min}$ will not move to machine 2, we have $\ell_1 = a_2 + c_1 + c_2 \le d_2 + a_{\min}/s \le a_2/s + d_2 \le 1$.
Then we consider the case when $d_1 > 0$. To obtain the desired bound, we need to further separate the proof into 2 subcases depending on the relation of $a_{\min}$ and a_2:

(1-3-1) The case when there is only one job in a_2, i.e., $a_{\min} = a_2$.

$$\begin{aligned} \max \quad & \ell_1 = a_2 + c_1 + c_2 \\ s.t. \quad & c_2 + s \cdot d_1 \le 1 && (35) \\ & a_2/s + s \cdot c_1 + d_2 \le 1 && (36) \\ & a_2 + c_1 + c_2 \le d_1 + d_2 + a_2/s && (37) \\ & a_2/s \le d_1 \\ & s \ge 1. && (38) \end{aligned}$$

Combining these constraints (35)–(38) with proper weights, we get:

$$2s \cdot (35) + 2 \cdot (36) + (37) + (2s^2 - 1) \cdot (38)$$
$$\Rightarrow (2s+1)(a_2 + c_1 + c_2) + d_2 \le 2s + 2$$
$$\Rightarrow \ell_1 = a_2 + c_1 + c_2 \le \frac{2s+2}{2s+1} \le \frac{4}{3}.$$

(1-3-2) The case when there are at least two jobs in a_2, i.e., $a_{\min} \le a_2/2$.

$$\begin{aligned} \max \quad & \ell_1 = a_2 + c_1 + c_2 \\ s.t. \quad & c_2 + s \cdot d_1 \le 1 && (39) \\ & a_2/s + s \cdot c_1 + d_2 \le 1 && (40) \end{aligned}$$

$$\begin{aligned} & a_2 + c_1 + c_2 \leq d_1 + d_2 + a_{\min}/s \leq d_1 + d_2 + a_2/2s \\ & s \geq 1. \end{aligned} \tag{41}$$

Combining these constraints (39)–(41) with proper weights, we get:

$$\begin{aligned} & \frac{1}{s} \cdot (39) + \frac{1+\sqrt{17}}{4} \cdot (40) + (41) \\ & \Rightarrow \left(\frac{\sqrt{17}-1}{4s} + 1\right) a_2 + \left(\frac{\sqrt{17}+1}{4} s + 1\right) c_1 + \left(1 + \frac{1}{s}\right) c_2 \\ & \quad + \left(\frac{\sqrt{17}-3}{4}\right) d_2 \leq \frac{\sqrt{17}+1}{4} + \frac{1}{s}. \end{aligned}$$

Since the coefficient of a_2 is greater than those of c_1 and c_2, and the coefficient of d_2 is greater than 0, we can obtain that:

$$\ell_1 = a_2 + c_1 + c_2 \leq \frac{\frac{\sqrt{17}+1}{4} + \frac{1}{s}}{\frac{\sqrt{17}-1}{4s} + 1} = \frac{\sqrt{17}+1}{4} < \frac{4}{3}.$$

(2) *When job $a_{\min}$ moves to machine 2, it is not the last processed job on machine 2, i.e., there is at least one job in $d_1 + d_2$ that is processed after job $a_{\min}$.*

Denote by d_s the total processing time of the jobs that will be processed after job $a_{\min}$ on machine 2 if job $a_{\min}$ moves to machine 2. Suppose the original allocation x is a NE. We claim that if the jobs of d_s are removed, the remaining allocation x' is still a NE. We prove this claim in the following.

First, we know that $d_1 + d_2 - d_s > c_1 + c_2$; Otherwise, job $a_{\min}$ can benefit by moving to machine 2. Thus, no job in $c_1 + c_2$ can benefit by moving to machine 2 even if removing the jobs of d_s. Second, since job $a_{\min}$, the job with smallest processing time in $a_1 + a_2$, will be processed before the jobs of d_s, removing the jobs of d_s will not change the completion time of any job in $a_1 + a_2$ on machine 2. Thus, no job in $a_1 + a_2$ can benefit by moving to machine 2 even if removing the jobs of d_s. At last, the jobs in machine $d_1 + d_2$ will also have no incentive to move to machine 1 after removing the jobs of d_s. Therefore, we prove that allocation x' is still a NE. Since allocation x' is the case when job $a_{\min}$ is the last job if moving to machine 2, the bound $\ell_1 \leq \frac{4}{3}$ of the previous case also holds in this case, which completes the proof of this lemma. □

Lemmas 13–15 yield the upper bound of the PoA of the FF-LPT policy. We show that the upper bound is tight in the following lemma:

Lemma 16 *The* PoA *of the FF-LPT policy is at least* $\frac{4}{3}$.

Proof Consider an instance where $a_1 = \frac{2}{3}$, $c_1 = \frac{1}{3} - \epsilon$, $c_2 = \frac{1}{3}$, $d_2 = \frac{2}{3}$, $a_2 = b_1 = b_2 = d_1 = 0$ and $s = \frac{1}{1-\epsilon}$, where ϵ is a small positive constant. The jobs on machine

1 (in the FF-LPT order) are: c_2, c_1 and a_1. The only job on machine 2 is d_2. Therefore, $\ell_1 = \frac{4}{3} - \epsilon$ and $\ell_2 = \frac{2}{3}$.

Obviously, jobs c_1, c_2 and d_2 have no incentive to move. The completion time of job a_1 is $\ell_1 = \frac{4}{3} - \epsilon$. If job a_1 moves to machine 2, its processing time will be $\frac{2}{3}/s = \frac{2}{3}(1-\epsilon)$, which is less than $\frac{2}{3}$. Thus, if job a_1 moves to machine 2, it will be processed after job d_2 and have the completion time $\frac{2}{3} + \frac{2}{3}(1-\epsilon) = \frac{4}{3} - \frac{2}{3}\epsilon$, which is greater than the completion time $\frac{4}{3} - \epsilon$ on machine 1. Therefore, no job can benefit by moving to the other machine, which means the instance is a NE.

In the optimal allocation, $\ell_1^* = a_1 + c_2 = 1$ and $\ell_2^* = d_2 + s \cdot c_1 = 1 - \frac{2}{3}\epsilon$. Therefore, we have that the PoA $\geq \frac{4}{3} - \epsilon$. This lemma holds when $\epsilon \to 0$. □

Lemmas 13–16 yield the following theorem:

Theorem 17 *The* PoA *of the FF-LPT policy is* $\frac{4}{3}$.

The above results also indicate the the bound for SPoA of FF-LPT policy. In the instance considered in Lemma 16, machine 2 has only a good job on it. Obviously, the job will not move to machine 1 in any circumstance. However, according to Proposition 22, if the instance is not a SE, the jobs on machine 2 must be in a coalition where every job can reduce its cost by moving to the other machine. Therefore, it holds that the instance in Lemma 16 is also a SE, and the SPoA of the FF-LPT policy is at least $\frac{4}{3}$. Since SPoA $\leq$ PoA by definition, we obtain following theorem:

Theorem 18 *The* SPoA *of the FF-LPT policy is* $\frac{4}{3}$.

4.2 Convergence to pure Nash equilibria

Theorem 19 *For the FF-LPT policy, best responses of jobs converge to a NE after n rounds in which jobs may make one selfish move each, in arbitrary order.*

Proof We will show that after round t, there will be at least t jobs that achieve their minimum possible completion time. Notice that any job who has achieved its minimum possible completion time will not change its strategy in the following rounds. Thus, best responses of jobs converge to a NE after at most n rounds.

Let J_t be the set of jobs that have achieved their minimum possible completion time after round t. We prove by induction that each round will have at least one more job that achieves its minimum possible completion time than last round, i.e., $|J_t| \geq |J_{t-1}| + 1$ for each round t.

We know that before round t, the jobs in J_{t-1} have already achieved their minimum possible completion time and will not change their strategies at round t. Let $\ell_1(J_{t-1})$ be the total processing time of the jobs in J_{t-1} who choose machine 1, and $\ell_2(J_{t-1})$ be that of the jobs in J_{t-1} who choose machine 2. Without loss of generality, suppose $\ell_1(J_{t-1}) \leq \ell_2(J_{t-1})$. On one hand, if there exist some jobs in $J - J_{t-1}$ whose favorite machine is machine 1, denote the one with maximum processing time among them by j_t. Obviously, in round t, the best possible choice for job j_t is machine 1 whatever strategies the other jobs in $J - J_{t-1}$ may take, because job j_t will be processed before any job in $J - J_{t-1}$ on machine 1. On the other hand, if all the jobs in $J - J_{t-1}$ have favorite machine as machine 2, denote the one with maximum processing time by j_t. We know that job j_t will be processed before any job in $J - J_{t-1}$ on any machine, because all the jobs in $J - J_{t-1}$ have favorite machine as machine 2 and j_t has the largest processing time. Thus, job j_t has only one best possible choice whatever strategies the other jobs in $J - J_{t-1}$ may take in round t. Because in either case, job j_t has the only one best possible choice in round t, job j_t will achieve its minimum possible completion time by best respond in round t. Therefore, we have $|J_t| \geq |J_{t-1}| + 1$. □

5 Experimental results

In this section, we show the performance of our FF-LPT policy by numerical simulations, in comparison with other classical policies. Specifically, the policies are:

1. The LPT (Longest Processing Time) policy: each machine processes all the jobs on the machine in descending order of the processing times of the jobs.
2. The SPT (Shortest Processing Time) policy: each machine processes all the jobs on the machine in ascending order of the processing times of the jobs.
3. The FF-SPT (Favorite First-Shortest Processing Time) policy: on each machine, the good jobs are processed before the bad jobs, and the good jobs and bad jobs are ordered by SPT rule, respectively.
4. The Makespan policy: each machine processes all jobs on the machine in parallel, and so the completion time of a job is the makespan of the machine that processes the job.

We generate random instances to test the five policies for different number of jobs n and different speed ratios s. For given n and s, we generate 30 random instances. In the instances, each job has an arbitrary favorite machine and a random size uniformly distributed in the interval [0, 100].

For each instance, we calculate the ratio between the cost of the equilibrium resulted by each policy and the optimal cost. The resulted equilibrium of each policy for each instance is achieved by the following approach: (1) generate a random allocation; (2) if there are some jobs that have the incentive to move, move the one with a smallest index; (3) repeat step (2) until no job has the incentive to move. Because the initial allocation generated in step (1) is randomized, we repeat the above approach for 30 times and take the mean cost of the 30 resulted equilibria as the final cost of the policy. As a different initial allocation may take a different number of moves to reach a equilibrium, we also use the average number of moves of the 30 initial allocations to denote the final number of moves to reach a equilibrium for the policy.

For simplicity, we use a lower bound instead of the optimal cost, since finding an optimum is NP-hard. Thus the ratios in the tables are slightly larger (worse) than the real ratios. The lower bound on the optimal cost is obtained by considering the following *fractional* assignment. First, allocate all jobs to their own favorite machines. Then, reassign a fraction of them to make all machines to have the same load. Thus, the

Table 4 The ratio between the cost of the NE resulted by each policy and the optimal cost

s	n	FF-LPT	FF-SPT	SPT	LPT	Makespan
1.1	5	1.061	1.182	1.220	1.075	1.100
	10	1.021	1.099	1.130	1.046	1.057
	15	1.016	1.070	1.080	1.059	1.052
	20	1.006	1.053	1.060	1.057	1.045
1.5	5	1.090	1.194	1.255	1.205	1.193
	10	1.019	1.116	1.126	1.261	1.175
	15	1.012	1.068	1.075	1.347	1.194
	20	1.009	1.062	1.068	1.383	1.202
2	5	1.090	1.235	1.235	1.436	1.322
	10	1.025	1.101	1.102	1.635	1.330
	15	1.013	1.066	1.065	1.694	1.349
	20	1.006	1.065	1.062	1.725	1.365
5	5	1.052	1.084	1.084	2.081	1.641
	10	1.022	1.073	1.073	3.214	1.944
	15	1.014	1.048	1.048	3.924	2.137
	20	1.009	1.048	1.048	3.832	2.171

lower bound is $opt \geq S_2 + \frac{s}{s+1}(S_1 - S_2)$ where $S_1 = \max\left\{\sum_{i:f_i=1} s_i,\ \sum_{i:f_i=2} s_i\right\}$ and $S_2 = \min\left\{\sum_{i:f_i=1} s_i,\ \sum_{i:f_i=2} s_i\right\}$.

For each instance, we calculate the mean cost of the equilibria and the optimal cost. We generate 30 random instances for testing each case of a speed ratio s and a number of jobs n. We consider cases for speed ratios $s = \{1.1,\ 1.5,\ 2,\ 5\}$, and for each s consider $n = \{5,\ 10,\ 15,\ 20\}$ as shown in Tables 4 and 5.

As we can see in Table 4, the ratios for our FF-LPT policy dominate the ratios for any other policies, and are very close to 1 (close to the optimum) in all the cases. The FF-SPT and the SPT policies are second to the FF-LPT policy, and the two policies have similar performance. However, the LPT policy and the Makespan policy perform badly especially for a larger speed ratio s. It worth noting that, the above numerical results are consistent with the theoretical results (shown in Table 1). In the theoretical results, our FF-LPT policy has a theoretical guarantee of $\frac{4}{3}$, while the PoAs for the FF-SPT and the SPT policies are both at least 1.618. The LPT and the Makespan policies, however, have no bounded guarantee when s is unbounded.

Table 5 shows the average number of moves to reach a equilibrium for each policy. The results indicate that our FF-LPT policy can converge to a equilibrium fast. However, it takes more moves for the LPT policy to converge to a equilibrium compared with other policies.

The numerical results reveal some insights for practical problems:

1. The FF-LPT policy tops other policies in both theory (worst case) and numerical results (average case). Thus the policy could be a good choice for practical problems.

Table 5 Number of moves to converge to a NW for each policy

s	n	FF-LPT	FF-SPT	SPT	LPT	Makespan
1.1	5	3.1	3.1	4.3	3.6	1.5
	10	7.9	7.2	11.9	14.6	2.7
	15	14.2	11.7	19.5	30.0	3.4
	20	16.3	14.4	33.1	48.8	3.6
1.5	5	3.0	2.8	3.3	4.3	1.9
	10	7.2	6.4	8.7	14.0	2.7
	15	11.8	9.8	11.7	25.2	3.7
	20	20.7	14.9	15.8	47.3	4.0
2	5	3.0	2.7	2.9	4.5	1.8
	10	7.0	5.7	6.9	12.3	3.1
	15	13.5	9.9	9.7	22.1	3.7
	20	18.0	13.2	14.4	36.7	4.3
5	5	2.7	2.6	2.4	3.2	2.8
	10	6.3	5.0	5.0	11.6	4.8
	15	9.4	7.6	7.5	23.5	6.3
	20	13.9	10.4	10.3	36.3	8.0

2. For small s (close to identical machines), the simple LPT policy has good enough performance. However, when s gets larger, the LPT policy performs poorly, and the measure of processing the more efficient jobs primarily (like the FF-LPT policy) can significantly improve the quality of equilibria.
3. In the perspective of the convergence time to a equilibrium, the Makespan policy is faster than the other policies. In a sense, the makespan policy can somewhat reduce the number of strategic behaviors of the users.

6 Conclusion and open questions

In this work, we analyze both the SPoA and PoA for the Makespan policy and design a new policy, called FF-LPT, which is better than several other classical policies for the favorite machine model. Specifically, we provide *exact* bounds on the PoA and the SPoA for *all values of* s, which allows us to compare with the same bounds for the two related machines (see Fig. 1). Since the classical policies (e.g., Makespan, LPT, SPT) have poor performance in terms of PoA, we propose the FF-LPT policy for this problem. We prove tight bounds on the PoA and the SPoA for the FF-LPT policy, and further study the convergence to and the existence of pure NE. Finally, we show through experimental results that the FF-LPT policy outperforms the other classical policies in terms of the average cases, which is consistent with our theoretical results. The numerical results also reveal some insights for choosing scheduling policies for practical applications. For example, for the system with larger speed ratio s, i.e., the speeds for a machine to process different types of jobs vary significantly, the measure of processing the more efficient type of jobs primarily (like the FF-LPT policy) can significantly improve the quality of equilibria.

To further the research in this paper, it would be interesting to extend the analysis to more machines in the *favorite* machines setting. Chen and Xu (2019) have analyzed the SPoA for the m machines case. One could also study coordination mechanisms for the problem for more machines. Considering other well studied solution concepts would also be interesting, including *sequential* PoA (Leme et al. 2012; Bilò et al. 2015; de Jong and Uetz 2014; Giessler et al. 2016), *approximate* SPoA (Feldman and Tamir 2009), and the *price of stochastic anarchy* (Chung et al. 2008).

Acknowledgements This work was supported by the National Natural Science Foundation of China (Nos. 71601152, 71720107002); the National Natural Science Foundation of China-Guangdong Joint Fund (No. U1901223); and the Natural Science Foundation of Guangdong Province (No. 2017A030312001).

A Postponed proofs

A.1 Proofs of Lemmas 2–6

Proof of Lemma 2 Since $a_1 > 0$, we consider four subcases: Case 2.1 ($a_2, c_1 = 0$), Case 2.2 ($a_2 = 0, c_1 > 0$), Case 2.3 ($a_2 > 0, c_1 = 0$) and Case 2.4 ($a_2, c_1 > 0$), which correspond to rows 2.1 to 2.4-b in Table 3.

The bound for Case 2.3 is the minimum of the bounds of 2.3-a, 2.3-b and 2.3-c. Similarly, the bound for Case 2.4 is the minimum of the bounds of 2.4-a and 2.4-b. Finally, the maximum of the bounds of the four subcases give the bound for this lemma. These 4 subcases are summed up in Fig. 4b (orange line). □

Proof of Lemma 3 Since $a_1 = a_2 = 0$, we consider three subcases: Case 3.1 ($c_1 = 0, c_2 > 0$), Case 3.2 ($c_1 > 0, c_2 = 0$) and Case 3.3 ($c_1 > 0, c_2 > 0$), which correspond to rows 3.1 to 3.3-b in Table 3.

The bound for Case 3.3 is the minimum of the bounds of 3.3-a and 3.3-b. Finally, the maximum of the bounds of the three subcases give the bound for this lemma. These 3 subcases are summed up in Fig. 4c (orange line). □

Proof of Lemma 4 Since $a_1 = 0$, $a_2, c_1 > 0$, we consider two subcases: Case 4.1 ($c_2 = 0$) and Case 4.2 ($c_2 > 0$), which correspond to rows 4.1-a to 4.2-c in Table 3.

The bound for Case 4.1 is the minimum of the bounds of 4.1-a and 4.1-b, where 4.1-b is the maximum of the bounds of 4.1-b.(8) and 4.1-b.(9). Note that it only needs one of constraints (8) and (9) holds, thus we take the maximum of 4.1-b.(8) and 4.1-b.(9). The bound for Case 4.2 is the minimum of the bounds of 4.2-a, 4.2-b, 4.2-c, where 4.2-a is the maximum of the bounds of 4.2-a.(14) and 4.2-a.(15). Finally, the maximum of the bounds of the two subcases give the bound for this lemma. These 2 subcases are summed up in Fig. 4d. □

Proof of Lemma 5 This lemma considers the case $a_1 = c_1 = c_2 = 0$ and $a_2 > 0$, which correspond to rows 5-a to 5-b.(9) in Table 3.

Similar as the above proofs, the bound for this case is the minimum of the bounds of 5-a and 5-b, where 5-b is the maximum of the bounds of 5-b.(8) and 5-b.(9). Furthermore, the bound of 5-b.(8) is the minimum of the bounds of 5-b.(8)-a and 5-b.(8)-b, where 5-b.(8)-b is the maximum of 5-b.(8)-b.(12) and 5-b.(8)-b.(13). Moreover 5-b.(8)-b.(12) is the minimum of 5-b.(8)-b.(12)-a and 5-b.(8)-b.(12)-b, where 5-b.(8)-b.(12)-b is the maximum of 5-b.(8)-b.(12)-b.(10) and 5-b.(8)-b.(12)-b.(11). Finally, the bounds give the bound for this lemma, which is shown in Fig. 4e. □

Proof of Lemma 6 This lemma consider the case $a_1 = c_1 = 0$ and $a_2, c_2 > 0$, which correspond to rows 6.(16)-a to 6.(17)-b.(9) in Table 3.

Similar as the above proofs, the bound for this case is the maximum of the bounds of 6.(16) and 6.(17), where 6.(16) is the minimum of 6.(16)-a, 6.(16)-b and 6.(16)-c, and 6.(17) is the minimum of 6.(17)-a and 6.(17)-b. Furthermore, the bound of 6.(16)-c is the maximum of the bounds of 6.(16)-c.(8) and 6.(16)-c.(9), where 6.(16)-c.(9) is the minimum of 6.(16)-c.(9)-a and 6.(16)-c.(9)-b where 6.(16)-c.(9)-b is the maximum of 6.(16)-c.(9)-b.(12) and 6.(16)-c.(9)-b.(13). Besides, the bound of 6.(17)-a is the maximum of bounds of 6.(17)-a.(10) and 6.(17)-a.(11), and 6.(17)-b is the maximum of 6.(17)-b.(8) and 6.(17)-b.(9). Finally, the bounds give the bound for this lemma, which is shown in Fig. 4f. □

A.2 Proof of lower bound (Lemma 7)

Table 2 gives instances that match all the bounds of Theorem 1. Each instance is represented by the setting of Fig. 2 and each symbol (e.g. $a_1, a_2, b_1 \dots$) represents at most one job.

Proposition 20 *The optimal makespan of each instance of Table 2 is at most 1.*

Proof It is easy to see that, for every instance of Table 2, the optimum schedule shown in Fig. 2 satisfies $\max\{\ell_1^*, \ell_2^*\} \leq 1$. Hence we can have $opt \leq 1$ even though the above schedule is not guaranteed to be a SE, because the optimal SE can only have better makespan than that schedule. □

Proposition 21 *Each schedule of Table 2 is a NE.*

Proof Because $\ell_1 \geq \ell_2$, no job in machine 2 can reduce its cost by moving to machine 1. Thus we only need to check that if there is no single job in machine 1 would benefit from moving to machine 2. Let ℓ_1' and ℓ_2' be the new loads of machine 1 and 2, respectively, after some movements of the jobs as stated in the corresponding context.

LB1 If a_2 moves to machine 2, $\ell_2' = \ell_2 + a_2/s = \frac{s^3+s^2+s+1}{s^3+2} = \ell_1$.

If c_2 moves to machine 2, $\ell_2' = \ell_2 + c_2 \cdot s = \frac{s^3+s^2+s+1}{s^3+2} = \ell_1$.

LB2 If a_2 moves to machine 2, $\ell_2' = \ell_2 + a_2/s = \frac{s^2+2s+1}{2s+1} = \ell_1$.

If c_2 moves to machine 2, $\ell_2' = \ell_2 + c_2 \cdot s = \frac{2s^2+2s}{2s+1} \geq \ell_1 = \frac{s^2+2s+1}{2s+1}$.

LB3 If a_2 moves to machine 2, $\ell_2' = \ell_2 + a_2/s = \frac{s+1}{s} = \ell_1$.

If c_2 moves to machine 2, $\ell_2' = \ell_2 + c_2 \cdot s = 2 \geq \ell_1 = \frac{s+1}{s}$.

LB4 If a_2 moves to machine 2, $\ell_2' = \ell_2 + a_2/s = \frac{s^3-s^2+2s-1}{s^3-s^2+s-1} = \ell_1$.

If c_2 moves to machine 2, $\ell_2' = \ell_2 + c_2 \cdot s = \frac{s^4-s^3+s^2-1}{s^3-s^2+s-1} \geq \ell_1 = \frac{s^3-s^2+2s-1}{s^3-s^2+s-1}$ by $s \geq s_3 \approx 1.755$.

LB5 If a_2 moves to machine 2, $\ell_2' = \ell_2 + a_2/s = \frac{3}{2} \geq \ell_1 = \frac{s+1}{2}$ by $s \leq s_5 = 2$.
If c_2 moves to machine 2, $\ell_2' = \ell_2 + c_2 \cdot s = \frac{s+2}{2} \geq \ell_1 = \frac{s+1}{2}$.
LB6 If a_2 moves to machine 2, $\ell_2' = \ell_2 + a_2/s = \frac{s^2-s+1}{s^2-s} = \ell_1$.
If c_2 moves to machine 2, $\ell_2' = \ell_2 + c_2 \cdot s = s \geq \ell_1 = \frac{s^2-s+1}{s^2-s}$ by $s \geq s_5 = 2$.
LB7 If a_2 moves to machine 2, $\ell_2' = \ell_2 + a_2/s = \frac{s^2}{2s-1} = \ell_1$.
LB8 If a_2 moves to machine 2, $\ell_2' = \ell_2 + a_2/s = 2 \geq \ell_1 = \frac{s+1}{s}$. □

According to Proposition 21 we know that no single job can reduce its cost by moving to other machine. It also holds that

Proposition 22 Epstein 2010 *Given a schedule on two machines which is a NE, if this schedule is not a SE, then a coalition of jobs where every job can reduce its cost consists of at least one job of each one of the machines.*

Proposition 23 *If a schedule of Table 2 is not a SE, then a coalition of jobs where every job can reduce its cost must consist of job a_2.*

Proof Because good job will become bad job after swap, if the coalition consists of only good jobs, it holds that $\ell_1' + \ell_2' \geq \ell_1 + \ell_2$. Thus at least one of the cost of the jobs will get worse after swap. Therefore, if a schedule is not a SE, then a coalition of jobs where every job can reduce its cost must consist of at least one bad job.

For LB7 and LB8, a_2 is in the coalition for sure, since a_2 is the only good job. For other instances in Table 2, if a_2 is not in the coalition then b_2 must in it according to Proposition 22. Next we will show that $a_2 + b_2/s \geq \ell_2$ for every instance of them, so that if b_2 is in the coalition then b_2 will not benefit from moving to machine 1. Thus a_2 must be in the coalition. For LB1 to LB3, we have $a_2 \geq \ell_2$ thus $a_2 + b_2/s \geq \ell_2$ holds. For LB4, $a_2 + b_2/s = \frac{2s^2-s}{s^3-s^2+s-1} \geq \ell_2 = 1$ by $s \leq s_4$. For LB5, $a_2 + b_2/s = \frac{s}{2} + \frac{1}{2} \geq \ell_2 = 1$. For LB6, $a_2 + b_2/s = \frac{1}{s-1} + \frac{1}{s} \geq \ell_2 = 1$ by $s \leq s_6 \approx 2.154$. □

Proposition 24 *For any instance of Table 2, if a_2 is the only job of machine 1 that in a coalition of jobs, these jobs cannot reduce their costs at the same time.*

Proof According to Proposition 22, if a_2 is the only job of machine 1 that in a coalition of jobs, there must be some jobs of machine 2 in this coalition. Thus this proposition says there are no such a a_2-$\{*\}$ swap that all these jobs can benefit simultaneously, where $\{*\}$ is any subset of jobs of machine 2. We prove this for the 8 instances of Table 2 one by one.

LB1 In this case a_2 moves to machine 2 and c_2 stays, it holds that no job (or subset of jobs) in machine 2 can benefit from moving to machine 1, because

No a_2-b_2 swap: $\ell_1' = c_2 + b_2/s = \frac{s+2}{s^3+2} \geq \ell_2 = \frac{s^3+1}{s^3+2}$ by $s \leq s_1 \approx 1.325$;
No a_2-d_1 swap: $\ell_1' = c_2 + d_1 \cdot s = \frac{s^3+1}{s^3+2} = \ell_2$;
No a_2-d_2 swap: $\ell_1' = c_2 + d_2 \cdot s = \frac{s^4-s^3-s^2+3s+1}{s^3+2} > \ell_2 = \frac{s^3+1}{s^3+2}$ by $s \leq s_1$.

LB2 In this case a_2 moves to machine 2 and c_2 stays, it holds that no job (or subset of jobs) in machine 2 can benefit from moving to machine 1, because

No a_2-b_2 swap: $\ell_1' = c_2 + b_2/s = 1 \geq \ell_2 = \frac{s^2+s}{2s+1}$ by $s \leq s_2 \approx 1.618$;
No a_2-d_2 swap: $\ell_1' = c_2 + d_2 \cdot s = \frac{s^2+s+1}{2s+1} > \ell_2 = \frac{s^2+s}{2s+1}$.

LB3 In this case a_2 moves to machine 2 and c_2 stays, it holds that no job (or subset of jobs) in machine 2 can benefit from moving to machine 1, because

No a_2-b_2 swap: $\ell_1' = c_2 + b_2/s = 1 = \ell_2$;
No a_2-d_2 swap: $\ell_1' = c_2 + d_2 \cdot s = \frac{1}{s} + 2s - s^2 > \ell_2 = 1$ by $s \leq s_3 \approx 1.755$.

LB4 In this case a_2 moves to machine 2 and c_2 stays, it holds that no job (or subset of jobs) in machine 2 can benefit from moving to machine 1, because

No a_2-b_2 swap: $\ell_1' = c_2 + b_2/s = 1 = \ell_2$;
No a_2-d_2 swap: $\ell_1' = c_2 + d_2 \cdot s = 1 = \ell_2$.

LB5 In this case a_2 moves to machine 2 and c_2 stay, it holds that no job (or subset of jobs) in machine 2 can benefit from moving to machine 1, because

No a_2-b_2 swap: $\ell_1' = c_2 + b_2/s = 1 = \ell_2$;
No a_2-d_2 swap: $\ell_2' = b_2 + a_2/s = \ell_1 = \frac{s+1}{2}$;
No a_2-$\{b_2, d_2\}$ swap: $\ell_1' = c_2 + b_2/s + d_2 \cdot s = \frac{-s^2+2s+2}{2} \geq \ell_2 = 1$.

LB6 In this case a_2 moves to machine 2 and c_2 stays, it holds that no job (or subset of jobs) in machine 2 can benefit from moving to machine 1, because

No a_2-b_2 swap: $\ell_1' = c_2 + b_2/s = 1 = \ell_2$.

LB7 In this case a_2 moves to machine 2, it holds that no job (or subset of jobs) in machine 2 can benefit from moving to machine 1, because

No a_2-d_1 swap: $\ell_1' = d_1 \cdot s = \frac{s(s-1)^2}{2s-1} \geq \ell_2 = \frac{s^2-s}{2s-1}$ by $s \geq s_6 \approx 2.154$;
No a_2-d_2 swap: $\ell_1' = d_2 \cdot s = \frac{s^2-s}{2s-1} = \ell_2$.

LB8 In this case a_2 moves to machine 2, it holds that no job (or subset of jobs) in machine 2 can benefit from moving to machine 1, because

No a_2-d_1 swap: $\ell_1' = d_1 \cdot s = 1 \geq \ell_2 = \frac{s^2-1}{s^2}$;
No a_2-d_2 swap: $\ell_1' = d_2 \cdot s = \frac{s^2-s-1}{s} \geq \ell_2 = \frac{s^2-1}{s^2}$ by $s \geq s_7 \approx 2.247$. □

Lemma 25 *Each schedule of Table 2 is a SE.*

Proof According to Propositions 23 and 24, we know that schedules of LB7 and LB8 are SE, since a_2 is the only job in machine 1. For LB1 to LB6, we will prove there is no such coalition of jobs can reduce their costs at the same time:

LB1 According to Proposition 23, we only need to consider the case both a_2 and c_2 of machine 1 are in the coalition. However, we have

No $\{a_2, c_2\}$-$\{b_2, d_2\}$ swap: $\ell_2' = d_1 + a_2/s + c_2 \cdot s = \frac{3s^2+2s-1}{s^3+2} \geq \ell_1 = \frac{s^3+s^2+s+1}{s^3+2}$ by $s \leq s_1$.

Since $\frac{s^2-1}{s^3+2} \leq \frac{s^3-s^2-s+2}{s^3+2} \leq \frac{s}{s^3+2}$ by $s \leq s_1$, i.e., $d_1 \leq d_2 \leq b_2$, we can know that a_2 and c_2 will not benefit even if the smallest job d_1 of machine 2 stays still. Thus all the jobs in machine 2 should be in the coalition. Since

No $\{a_2, c_2\}$-$\{b_2, d_1, d_2\}$ swap: $\ell_1' = b_2/s + d_1 \cdot s + d_2 \cdot s = \frac{s^4-s^2+s+1}{s^3+2} \geq \ell_2 = \frac{s^3+1}{s^3+2}$ by $s \leq s_1$,

we know that there is no such coalition of jobs that every job can reduce its cost by deviating simultaneously.

LB2 Similar as the former case, we consider the case both a_2 and c_2 are in the coalition. We get the same result by the fact that

No $\{a_2, c_2\}$-$\{*\}$ swap: $\ell_2' \geq a_2/s + c_2 \cdot s = \frac{s^2+2s+1}{2s+1} = \ell_1$.

LB3 Considering the case both a_2 and c_2 are in the coalition, we have that

No $\{a_2, c_2\}$-$\{*\}$ swap: $\ell_2' \geq a_2/s + c_2 \cdot s = \frac{s+1}{s} = \ell_1$.

LB4 Considering the case both a_2 and c_2 are in the coalition, we have that

No $\{a_2, c_2\}$-$\{*\}$ swap: $\ell_2' \geq a_2/s + c_2 \cdot s = \frac{s^4-ss^3+2s^2}{s^3-s^2+s-1} \geq \ell_1 = \frac{s^3-s^2+2s-1}{s^3-s^2+s-1}$ by $s \geq s_3 \approx 1.755$.

LB5 Considering the case both a_2 and c_2 are in the coalition, we have that

No $\{a_2, c_2\}$-$\{*\}$ swap: $\ell_2' \geq a_2/s + c_2 \cdot s = \frac{s+1}{2} = \ell_1$.

LB6 Considering the case both a_2 and c_2 are in the coalition, we have that

No $\{a_2, c_2\}$-$\{*\}$ swap: $\ell_2' \geq a_2/s + c_2 \cdot s = \frac{1}{s^2-s} + s - 1 \geq \ell_1 = \frac{s^2-s+1}{s^2-s}$ by $s \geq s_5 = 2$. □

A.3 Proofs of Lemmas 9–12

Proof of Lemma 9 Rows 2.1, 2.2 and 3.4-a in Table 3 cover all the cases. Since each of the 3 bounds is below the claimed bound $\frac{s^3+s^2+s+1}{s^2+s+1}$, the lemma is proved. □

Proof of Lemma 10 Similar to the proof of SPoA, according to rows 10.1-a to 10.2 of Table 3 we have

$$\max\left\{\min\left\{\frac{2(s+1)}{s+2}, s\right\}, s\right\} \leq \frac{s^3+s^2+s+1}{s^2+s+1}$$

for $s \geq 1$, thus the lemma is proved. □

Proof of Lemma 11 The bound of row 11 of Table 3 implies this lemma. □

Proof of Lemma 12 We know that

$$\ell_1 = a_2 + c_2 = \frac{s^3+s^2+s+1}{s^2+s+1},$$

$$\ell_2 = b_1 + b_2 = \frac{s^3+1}{s^2+s+1}.$$

If a_2 goes to machine 2, the cost of a_2 becomes

$$\ell_2' = \ell_2 + a_2/s = \frac{s^3+s^2+s+1}{s^2+s+1}.$$

Similarly, if c_2 goes to machine 2, the cost of c_2 becomes

$$\ell_2' = \ell_2 + c_2 \cdot s = \frac{s^3+s^2+s+1}{s^2+s+1}.$$

Thus the schedule is a NE. □

References

Andelman N, Feldman M, Mansour Y (2009) Strong price of anarchy. Games Econ Behav 2(65):289–317

Aumann RJ (1959) Acceptable points in general cooperative n-person games. Princeton University Press, Princeton, pp 287–324

Awerbuch B, Azar Y, Richter Y, Tsur D (2006) Tradeoffs in worst-case equilibria. Theor Comput Sci 361(2):200–209

Azar Y, Fleischer L, Jain K, Mirrokni V, Svitkina Z (2015) Optimal coordination mechanisms for unrelated machine scheduling. Oper Res 63(3):489–500

Bilò V, Flammini M, Monaco G, Moscardelli L (2015) Some anomalies of farsighted strategic behavior. Theory Comput Syst 56(1):156–180

Caragiannis I (2013) Efficient coordination mechanisms for unrelated machine scheduling. Algorithmica 66(3):512–540

Caragiannis I, Fanelli A (2019) An almost ideal coordination mechanism for unrelated machine scheduling. Theory Comput Syst 63(1):114–127. https://doi.org/10.1007/s00224-018-9857-2

Caragiannis I, Gkatzelis V, Vinci C (2017) Coordination mechanisms, cost-sharing, and approximation algorithms for scheduling. In: Web and internet economics—13th international conference, WINE 2017, Bangalore, 17–20 December 2017, Proceedings, pp 74–87

Chen B, Gürel S (2012) Efficiency analysis of load balancing games with and without activation costs. J Sched 15(2):157–164

Chen C, Xu Y (2019) Selfish load balancing for jobs with favorite machines. Oper Res Lett 47(1):7–11

Chen L, Ye D, Zhang G (2014) Online scheduling of mixed CPU-GPU jobs. Int J Found Comput Sci 25(06):745–761

Chen Q, Lin L, Tan Z, Yan Y (2017) Coordination mechanisms for scheduling games with proportional deterioration. Eur J Oper Res 263(2):380–389

Chen C, Penna P, Xu Y (2020) Online scheduling of jobs with favorite machines. Comput Oper Res 116(104):868. https://doi.org/10.1016/j.cor.2019.104868

Cho Y, Sahni S (1980) Bounds for list schedules on uniform processors. SIAM J Comput 9(1):91–103

Christodoulou G, Koutsoupias E, Nanavati A (2009) Coordination mechanisms. Theor Comput Sci 410(36):3327–3336

Chung C, Ligett K, Pruhs K, Roth A (2008) The price of stochastic anarchy. In: Proc. of the 1st int. symp. on algorithmic game theory (SAGT), LNCS, vol 4997, pp 303–314

Czumaj A, Vöcking B (2007) Tight bounds for worst-case equilibria. ACM Trans Algorithms (TALG) 3(1):4

de Jong J, Uetz M (2014) The sequential price of anarchy for atomic congestion games. In: Proc. of the 10th international conference on web and internet economics (WINE), LNCS, vol 8877, pp 429–434

Epstein L (2010) Equilibria for two parallel links: the strong price of anarchy versus the price of anarchy. Acta Inform 47(7):375–389

Feldman M, Tamir T (2009) Approximate strong equilibrium in job scheduling games. J Artif Intell Res 36:387–414

Feldmann R, Gairing M, Lücking T, Monien B, Rode M (2003) Nashification and the coordination ratio for a selfish routing game. In: ICALP 2003, vol 30. Springer, pp 514–526

Fiat A, Kaplan H, Levy M, Olonetsky S (2007) Strong price of anarchy for machine load balancing. In: ICALP 2007, vol 4596. Springer, pp 583–594

Finn G, Horowitz E (1979) A linear time approximation algorithm for multiprocessor scheduling. BIT Numer Math 19(3):312–320

Giessler P, Mamageishvili A, Mihalák M, Penna P (2016) Sequential solutions in machine scheduling games. CoRR abs/1611.04159

Gonzalez T, Ibarra OH, Sahni S (1977) Bounds for lpt schedules on uniform processors. SIAM J Comput 6(1):155–166

Hassin R, Yovel U (2015) Sequential scheduling on identical machines. Oper Res Lett 43(5):530–533

Hoeksma R, Uetz M (2019) The price of anarchy for utilitarian scheduling games on related machines. Discrete Optim 31:29–39

Immorlica N, Li LE, Mirrokni VS, Schulz AS (2009) Coordination mechanisms for selfish scheduling. Theor Comput Sci 410(17):1589–1598

Koutsoupias E, Papadimitriou C (1999) Worst-case equilibria. In: Proceedings of the 16th annual symposium on theoretical aspects of computer science (STACS). Springer, pp 404–413

Kress D, Meiswinkel S, Pesch E (2018) Mechanism design for machine scheduling problems: classification and literature overview. OR Spectrum 40(3):583–611

Leme RP, Syrgkanis V, Tardos É (2012) The curse of simultaneity. In: Proc. of innovations in theoretical computer science (ITCS), pp 60–67

Nong Q, Fan G, Fang Q (2017) A coordination mechanism for a scheduling game with parallel-batching machines. J Combin Optim 33(2):567–579

Schuurman P, Vredeveld T (2007) Performance guarantees of local search for multiprocessor scheduling. INFORMS J Comput 19(1):52–63

Vakhania N, Hernandez JA, Werner F (2014) Scheduling unrelated machines with two types of jobs. Int J Prod Res 52(13):3793–3801

Ye D, Chen L, Zhang G (2019) On the price of anarchy of two-stage machine scheduling games. J Combin https://doi.org/10.1007/s10878-019-00474-2

Zhang L, Zhang Y, Du D, Bai Q (2019) Improved price of anarchy for machine scheduling games with coordination mechanisms. Optim Lett 13(4):949–959