Journal of Combinatorial Optimization (2021) 42:616–635
https://doi.org/10.1007/s10878-019-00474-2

# On the price of anarchy of two-stage machine scheduling games

**Deshi Ye**[1] · **Lin Chen**[2] · **Guochuan Zhang**[1]

Published online: 1 November 2019


## Abstract

We consider a scheduling game, in which both the machines and the jobs are players. Machines are controlled by different selfish agents and attempt to maximize their workloads by choosing a scheduling policy among the given set of policies, while each job is controlled by a selfish agent that attempts to minimize its completion time by selecting a machine. Namely, this game was done in two-stage. In the first stage, every machine simultaneously chooses a policy from some given set of policies, and in the second stage, every job simultaneously chooses a machine. In this work, we use the price of anarchy to measure the efficiency of such equilibria where each machine is allowed to use one of the at most two policies. We provide nearly tight bounds for every combination of two deterministic scheduling policies with respect to two social objectives: minimizing the maximum job completion, and maximizing the minimum machine completion time.

**Keywords** Price of anarchy · Scheduling · Coordination mechanisms

A preliminary version of this paper appeared in the 11th International Conference on Combinatorial Optimization and Applications (COCOA 2017,pp. 214–225). Research was supported in part by NSFC (11671355, 11771365) and NSF (1756014).

✉ Deshi Ye
yedeshi@zju.edu.cn

Lin Chen
chenlin198662@gmail.com

Guochuan Zhang
zgc@zju.edu.cn

[1] College of Computer Science, Zhejiang University, Hangzhou 310027, China

[2] Department of Computer Science, Texas Tech University, Lubbock, TX, USA

## 1 Introduction

This paper studies a scheduling game model that arose in cloud computing. Cloud provides an attractive platform for two entities: service providers (or machine owners) and users (or jobs). In our game model, both of them are players. Selfish Users want to minimize the completion time by choosing a proper machine, while the service providers attempt to attract more jobs by specifying a scheduling policy. A scheduling policy is an algorithm for the machine to schedule all the jobs that have been assigned to by the users.

Most previous games (Koutsoupias and Papadimitriou 1999; Immorlica et al. 2009) on scheduling consider only one-side, jobs or machines are the players, but not both. To the best of our knowledge, Ashlagi et al. (2010) were the first ones to study the model, in which both machines and jobs are selfish players. This game will give a set of scheduling policies at the beginning. In the first stage, every machine picks up a scheduling policy with the goal to maximize its load. In the second stage, every job simultaneously chooses a machine such that its completion time is minimized. The game reaches a *Nash equilibrium* if no machine would like to change its policy, and no job has the incentive to switch machines. Ashlagi et al. (2010) proved that there always exists a pure strategy Nash equilibrium if the machines are restricted to use two deterministic strategies. Besides, they have shown that there may not exist a pure strategy Nash equilibrium if the machines are allowed to use more than two deterministic policies.

It is worthy to note that a Nash equilibrium does not always get the optimal social welfare. Actually, selfish behavior might lead to highly inefficient outcome (Ashlagi et al. 2010). Moreover, there might exist many different equilibria. It is challenging to figure out the quality of such equilibria. The quality of a Nash equilibrium is measured with respect to the social optimum. In this work, we consider two social objectives: minimizing the maximum completion time of the jobs (we call it the *Min-Max* model), and maximizing the minimum machine completion time (we call it the *Max-Min* model).

To measure the efficiency of a game $G$ with respect to a social objective, we adopt the *price of anarchy* (PoA) or the *coordination ratio* that was introduced by Koutsoupias and Papadimitriou (1999). The price of anarchy has been extensively studied in many game-theoretic models, such as job scheduling (Nisan et al. 2007; Immorlica et al. 2009), selfish routing (Roughgarden and Tardos 2002), network formation (Fabrikant et al. 2003), facility location (Vetta 2002), congestion games (Roughgarden and Tardos 2004), greedy combinatorial auctions (Lucier and Borodin 2010; Feldman et al. 2016).

Let $NE(G)$ be the set of Nash equilibria in the game $G$. The social cost of a game $G$ is a function $C(v)$ for each $v \in NE(G)$ that numerically expresses the social objective of an outcome $v$ of the game. The social optimum $OPT(G)$ is the optimal value in the corresponding optimization problem. The price of anarchy of a game $G$ is the worst-case ratio over all equilibria to the social optimum. Formally, it is defined as

$$PoA(G) = \sup_{v \in NE(G)} \left\{ \frac{C(v)}{OPT(G)} \right\},$$

if the social objective is a minimization function. Similarly, for a maximization objective function, we have

$$PoA(G) = \sup_{v \in NE(G)} \left\{ \frac{OPT(G)}{C(v)} \right\}.$$

**Related work** To the best of our knowledge, there are not too many works on two-stage scheduling games. Recently, Chen et al. (2012) studied a two-stage scheduling game, in which machines can reject some of their jobs. The motivation for rejecting jobs is to attract more valuable jobs. Most scheduling games focus on the scenario that only the jobs are players, where every job attempts to switch machines to minimize its own completion time, where the completion time of a job refers to the load of the machine it is assigned to. Immorlica et al. (2009) proved that the prices of anarchy are $2-2/(m+1)$, $\Theta(\log m/\log\log m)$, and unbounded for identical parallel machine scheduling, related machine scheduling, and unrelated machine scheduling, respectively, where $m$ is the number of machines.

The price of anarchy is to measure the efficiency of equilibrium. To reduce the price of anarchy, *coordination mechanism* was first introduced by Christodoulou et al. (2004). The coordination mechanism for a game is a set of local policies, one for each machine, which will determine the completion time of a job that has been assigned to it.

The scheduling policies can be Makespan (M), ShortestFirst (S), LongestFirst (L), Randomized (R) (detailed definitions are given in Sect. 2). In contrast to our model, each machine in a coordination mechanism game does not change its policy during the whole game. Note that a coordination mechanism game with all machines using the policy Makespan (M) is exactly the classic scheduling game (Nisan et al. 2007). The motivation to use the Makespan policy is that all jobs in a machine will complete at the same time in some scenario. Here the notation of Makespan is a bit overused, sometimes it refers to the scheduling policy, sometimes it refers to the social objective. Anyway, we will point it out explicitly when we use it.

Let us consider the social objective is to minimize the makespan, namely the longest completion time of jobs, and we will introduce the prices of anarchy in various coordination mechanism games as below. For identical parallel machines, the PoA of the game with the policies Makespan, ShortestFirst, LongestFirst, Randomized are $2-2/(m+1)$ (Immorlica et al. 2009; Finn and Horowitz 1979), $2-1/m$ (Immorlica et al. 2009; Graham 1966), $4/3-1/(3m)$ (Graham 1969; Christodoulou et al. 2004), $2-2/(m+1)$ (Finn and Horowitz 1979), respectively, where $m$ is the number of machines. For related machines, the PoA of the game with the ShortestFirst policy is $\Theta(\log m)$ (Immorlica et al. 2009), the PoA of the game with the LongestFirst policy is at least 1.52 and at most 1.59 (Dobson 1984; Friesen 1987), the PoAs of the game with Makespan policy and Randomized policy are $\Theta(\log m/\log\log m)$ (Koutsoupias and Papadimitriou 1999). For unrelated machines, there are a number of results, see e.g. Azar et al. (2008), Caragiannis (2009), Immorlica et al. (2009). We refer to the survey (Heydenreich et al. 2007; Nisan et al. 2007) for the study of selfish scheduling.

There are also many studies of scheduling games on the social objective of maximizing the minimum machine load. Deuermeyer et al. (1982) investigated a coordination

**Table 1** The PoAs of different games

| Model | (S, L) | (S, M) | (L, M) |
|---|---|---|---|
| Min-Max $m = 2$ | 9/7 | 3/2 | 7/6 |
| Min-Max $m \geq 3$ | $2m/(m+1)$ | $2-1/m$ | $2m/(m+1)$ |
| Max-Min $m \geq 2$ | $[2-2/(m-1), 2-1/m]$ | $m$ | [1.691, 1.7] |

mechanism scheduling game with all machines using the policy LongestFirst. They showed the upper bound is of at most $4/3-1/3m$ for identical parallel machines. The scheduling game with policy Makespan was studied in Epstein et al. (2009), Chen et al. (2013), it was shown that the PoA is bounded in 1.691 and 1.7. Furthermore, it was mentioned in Chen et al. (2013), Epstein et al. (2014) that the PoA was unbounded in the related machine model. Some restricted cases of the related machine scheduling game where the speed ratio is of at most two was studied in Epstein et al. (2014), Tan et al. (2012), Lin and Tan (2014).

Finally, we mention that there are some works on the social objective of minimizing the (or weighted) sum of completion time. Cole et al. (2011) showed that the PoA is 4 for unrelated machines. Hoeksma and Uetz (2011) studied the related machine scheduling under the SPT (shortest processing time first) rule, and presented an upper bound of 2 and a lower bound of $e/(e-1) \approx 1.58$.

**Our contribution** As indicated in Ashlagi et al. (2010), there might not exist an equilibrium if machines are allowed to use more than two policies. They claimed that there exists Nash equilibrium when machines are limited to use two deterministic policies. However, this claim is inaccurate. In this work, we first show the existence of Nash equilibrium under a necessary assumption, even if the machines are restricted to use only two policies. Next, we give a detailed analysis of the performance via the price of anarchy. We assume that the local policies for machines are limited to ShortestFirst, LongestFirst, and Makespan. We denote it as a $(S, L)$-game if the two policies are ShortestFirst and LongestFirst. Similarly, we define the $(S, M)$-game and the $(L, M)$-game, respectively. Table 1 summarizes the results of the three games, where a single number presents a tightly bound and an interval presents a lower bound and an upper bound.

In the remaining part of the paper, we first present the problem statement and settings in Sect. 2. Then we address the analysis of the price of anarchy on two social objectives in Sects. 3 and 4, respectively. We conclude the paper with open questions in Sect. 5.

## 2 The game settings

Let $\mathcal{M} = \{1, 2, \ldots, m\}$ be the set of identical machines, and $\mathcal{N} = \{1, 2, \ldots, n\}$ be the set of jobs. Both jobs and machines are selfish players. Each job $j$, associated with a processing time (or size) $a_j$, attempts to minimize his own completion time. Each

machine can select a local scheduling policy to maximize the workload, which is the total processing time of all the jobs that have been assigned to that machine.

Let $P_i$ be the set of all possible strategies of machine $i$, and let $P = P_1 \times P_2 \times \ldots \times P_m$ be the set of all possible machine strategy profiles. Let $p = (p_1, p_2, \cdots, p_i, p_{i+1}, \ldots, p_m)$ be a specific vector of machines' strategies, where $p_i \in P_i$. And let us denote $p_{-i}$ be the vector of all strategies except machine $i$'s strategy, *i.e.*, $p_{-i} = (p_1, p_2, \ldots, p_{i-1}, p_{i+1}, \ldots, p_m)$.

In this work, we consider three scheduling policies, namely ShortestFirst (S), LongestFirst (L), and Makespan (M). That is $P_i = \{S, L, M\}$ for all $i$. The *ShortestFirst* policy (the *LongestFirst* policy) executes jobs in the non-decreasing (non-increasing) order of processing times. The *Makespan* policy processes the jobs in a batch such that all the jobs complete at the same time, *i.e.*, the completion time of every job is the workload of the machine. Actually, every policy determines the priority of jobs that have been assigned to that machine. In ShortestFirst policy, a job with a shorter processing time has a higher priority. In LongestFirst policy, a job with a longer processing time has a higher priority. In Makespan policy, all jobs have the same priorities.

In the policies stated above, ties are broken in favor of the job with the lowest index. If two jobs have the same length, then the one with the lower index has a higher priority.

Let $Q_j$ be the set of all possible strategies of job $j$, and let $Q = Q_1 \times Q_2 \times \cdots \times Q_n$ be the set of all possible job strategy profiles. Then $Q_j = \mathcal{M}$ for all $j$. Let $q = (q_1, q_2, \ldots, q_j, q_{j+1}, \ldots, q_n)$ be a specific vector of jobs' strategies, where $q_j \in Q_j$. And let us denote $q_{-j}$ be the vector of all strategies except job $j$'s strategy, *i.e.*, $q_{-j} = (q_1, q_2, \ldots, q_{j-1}, q_{j+1}, \ldots, q_n)$.

Let $c_j^p(q)$ be the completion time of job $j$ with machine policy profile $p \in P$ and the job profile $q \in Q$. Let $l_i^q(p)$ be the workload of machine $i$ with job policies $q \in Q$ and machine policy $p \in P$.

**Definition 1** (*Job equilibrium*) A job action profile $q = (q_1, \ldots, q_n)$ is a pure Nash equilibrium in the game with policies $p = (p_1, p_2, \ldots, p_m)$ if for every job $j$, $c_j^p(q_j, q_{-j}) \leq c_j^p(q'_j, q_{-j})$ for every $q'_j \in Q_j$.

**Definition 2** (*Machine equilibrium*) A machine action profile $p = (p_1, \ldots, p_m)$ is a pure Nash equilibrium in the game with job policies $q = (q_1, q_2, \ldots, q_n)$ if for every machine $i$, $l_i^q(p_i, p_{-i}) \geq l_i^q(p'_i, p_{-i})$ for every $p'_i \in P_i$.

In our problem, the scheduling game can be stated in *two stages*. At the first stage, machines announce their scheduling policies. Then in the second stage, each job chooses a machine to minimize his completion time. A job will move to another machine if this can decrease his completion time. A job (or a machine) is *satisfied* if this job (or a machine) does not have an incentive to change its decision. Once all the jobs (or machines) satisfied, we called it a *job equilibrium* (or a machine equilibrium).

We say that the game reaches a pure Nash equilibrium if it is both in a job equilibrium and in a machine equilibrium. Namely, a Nash equilibrium is a solution in which no player (either job or machine) can improve his utility by changing only his own strategy.

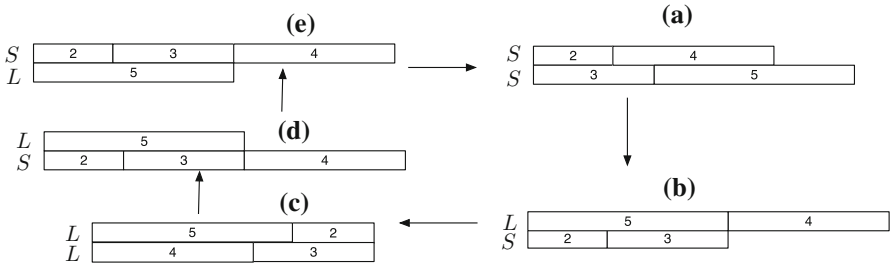


**Fig. 1** An example to illustrate that there is no Nash equilibrium if each machine will change its policy once it benefits in one job equilibrium

Ashlagi et al. (2010) claimed that they have proved that there exists a pure Nash equilibrium if the machines are restricted to use any two deterministic policies. However, this claim is inaccurate. Figure 1 illustrates a scenario that the algorithm in Ashlagi et al. (2010) did not find an equilibrium. In details, there are two machines and four jobs with processing times 2, 3, 4, 5, respectively. Each sub-figure shows a job equilibrium. Initially, both the machines use the policy S, illustrated in Fig. 1a. If the first machine changes its policy from S to L, Fig. 1b shows one possible job equilibrium. In this equilibrium, the first machine has workload 9, which is larger than the workload 6 in Fig. 1a, which indicates that the first machine will change its policy from S to L. Similarly, one will get a job equilibrium from Fig. 1a–d. We rename the machines, and then Fig. 1d becomes 1e. Thus this is a loop demonstrating the change of machine policies. Along the loop, there is always a job equilibrium so that exactly one machine can get better by changing the current policy.

Note that the problem arises in the above example is that given a profile of the machines, there might exist several job equilibria. To find a Nash equilibrium for our problem, we need to make an assumption that a machine has an incentive to alter his policy if and only if his workload will strictly increase in all the resulting job equilibria after the change. Violating this assumption, the existence of a Nash equilibrium cannot be guaranteed. This example reaches a Nash equilibrium even under this assumption. As we know, Fig. 1b, d are both job equilibria when one machine in Fig. 1a changes his policy from S to L. In the Fig. 1d, the load of the machine with policy L is 5, which is smaller than the equilibrium in Fig. 1a with load of 6 or 8. Therefore, no machines in Fig. 1a have the incentive to change their policies.

Under the assumption, we define a Nash equilibrium for our two-stage scheduling game as below.

**Definition 3** (*Nash equilibrium*) A profile $(p, q)$ is a pure Nash equilibrium if

1. $q$ is a pure job Nash equilibrium with $p$, and
2. for any machine $i$, $l_i^q(p_i, p_{-i}) \geq l_i^{q'}(p_i', p_{-i})$ holds for every $(q', p_i') \in Q \times P_i$ such that $q'$ is a pure job Nash equilibrium with $(p_i', p_{-i})$.

**Definition 4** (*Consistency*) A scheduling policy is consistent if for every two jobs $i$, $j$ $(i \neq j)$ where they are both assigned on the same machine, job $i$ is always scheduled before job $j$, or job $j$ is always scheduled before job $i$.

**Remark** The policy *ShortestFirst*, the policy *LongestFirst*, and the policy *Makespan* are consistent.

In the following, we will show the existence of a pure Nash equilibrium.

**Theorem 1** *In the two-stage game, there exists a pure Nash equilibrium, if machines are only allowed to use two arbitrary deterministic and consistent policies, and a machine has an incentive to change its policy only if its workload will strictly increase in all resulting job equilibria after the change.*

***Proof*** Ashlagi et al. (2010) have shown that there exists a job equilibrium if we fix a profile of machines using these two deterministic and consistent policies. We call it a *round* of the game if we fix a profile of machines, and then there reaches a job equilibrium due to this machine profile. Let $\psi, \phi$ be the two deterministic and consistent policies that a machine might adopt.

The proof of this Theorem can be done by construction. We will show the following routine finds a Nash equilibrium. For any given profile, we change the profile of the machines one by one at each round, *i.e.*, only one machine will change its policy at a round. In particular, we show that if a machine switches one policy to another in one round, then it will never switch back in the next round. At round $t$ of the game, let $x(t)$ be the number of machines use the policy $\psi$, and $y(t)$ be the number of machines use the policy $\phi$, where $x(t) + y(t) = m$. Then we claim that either $x(t)$ or $y(t)$ is a non-decreasing function of $t$.

If none of the machines has the incentive to change policy, *i.e.*, $x(t+1) = x(t)$, then the theorem follows immediately. Thus let us consider $x(t+1) = x(t) + 1$ or $x(t+1) = x(t) - 1$. W.L.O.G., we only prove that $x(t)$ is a non-decreasing function of $t$ when $x(t+1) = x(t) + 1$. Otherwise, $y(t)$ is a non-decreasing function of $t$.

At round $t+2$, we can show that no machine with policy $\psi$ changes to $\phi$, *i.e.*, $x(t+2) \geq x(t+1)$. Otherwise, we shall have $x(t+2) = x(t)$. We have illustrated in round $t+1$ there exists a job equilibrium in which one machine uses the policy $\phi$ can increase his load by switching his policy. In other words, there exists a job equilibrium such that one machine of $\psi$ will decrease the load after changing his policy. Hence, we got a contradiction.

In sums, the game runs at most $m$ rounds, and then finally reaches a Nash equilibrium. □

## 3 The PoA in the Min-Max model

In this section, we study the game with the social objective to minimize the makespan. The selfish actions of machines may result in high social costs, because the machines attempt to increase their workloads. We consider the $(S, M)$-game, the $(S, L)$-game, and the $(L, M)$-game, respectively.

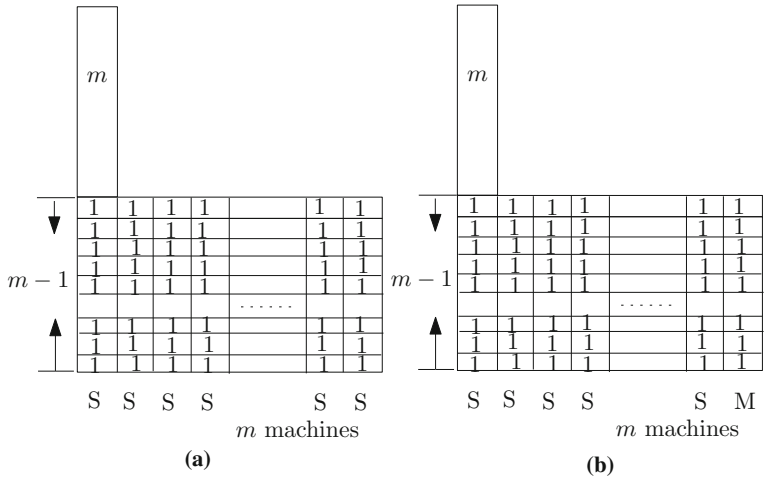


**Fig. 2** Job configuration in the profile $(S, M)$

### 3.1 The (*S*, *M*)-game

In this game, the machines are only allowed to use either the ShortestFirst policy or the Makespan policy. The idea of analysis is similar to the list scheduling (Graham 1966) by Graham.

**Theorem 2** *The price of anarchy of the* $(S, M)$*-game is* $2 - 1/m$.

***Proof*** Consider any Nash equilibrium NE of the game. Let $L_j$ be the workload of machine $j$ in the equilibrium NE. Assume the makespan is determined by machine $i$, *i.e.*, $L_i = \max_j L_j$. Let $y$ be the processing time of the last job on machine $i$. Clearly $L_i - L_j \leq y$ for any $j$, as the schedule is an equilibrium. Recall that $C^{OPT}$ and $C^{NE}$ are respectively the social cost of the optimal solution and the cost of the equilibrium. We have $C^{OPT} \geq L_i - y + y/m$, *i.e.*,

$$C^{NE} = L_i \leq C^{OPT} + (1 - 1/m)y. \tag{1}$$

It implies an upper bound of $2 - 1/m$ since $y \leq C^{OPT}$.

The following instance that we borrowed from the lower bound for list scheduling (Graham 1966) shows the tightness of the bound. There are $m(m-1)$ small jobs of unit processing time and one large job of processing time $m$. Figure 2a shows a job equilibrium that all machines use the policy S. Clearly, machine 1 (the leftist machine) will not change his policy. Figure 2b illustrates a job equilibrium when one of the machines (other than the first one) changes his policy to M. The equilibrium shows that this machine does not strictly increase his load by changing policy. Thus, the PoA is at least $2 - 1/m$. □

### 3.2 The (*S*, *L*)-game

We show that the price of anarchy for the $(S, L)$-game is $2m/(m+1)$ for $m \geq 3$, while it becomes $9/7$ for $m = 2$.

**Theorem 3** *The price of anarchy of the* $(S, L)$*-game is at most* $2m/(m+1)$*, for* $m \geq 3$.

***Proof*** Denote by $\Phi$ the set of machines that use the policy S, and $\Psi$ the set of machines that use the policy L, respectively. Note that $\Phi$ can be empty (or $\Psi$ is empty) which represents that all the machines use the same policy $L$ (or $S$).

We consider the job $k$ with processing time $y$, which is the last job on machine $i$ that determines the makespan. Denote $L_i$ to be the load of machine $i$. Let $x = L_i - y$ be the load of the machine $i$ without job $k$. Again, denote $C^{OPT}$ and $C^{NE}$ to be the social cost of the optimal solution and the cost of the equilibrium, respectively.

One can easily check that the theorem follows immediately if $y \leq m/(m+1)C^{OPT}$. In the following we assume that $y > m/(m+1)C^{OPT}$ and the proof is done by contradiction. Actually, we can also assume that $y + x > 2m/(m+1)C^{OPT}$, since otherwise the theorem follows trivially. Now we have $\max\{y, x\} > m/(m+1)C^{OPT}$.

**Case 1** Machine $i$ uses the policy $S$, *i.e.*, $i \in \Phi$. We know that the load of each machine in $\Phi$ other than $i$ is not less than $y$. Otherwise, this machine will change its policy to L and its load will be at least $y$, which contradicts with the fact that the current schedule is an equilibrium. We also note that $L_j \geq x$ from the point that job $k$ does not move to machine $j$ for any $j \neq i$. On the other hand, the load of each machine in $\Psi$ is at least $y$, otherwise, the job $k$ will move that machine. Hence, we have $L_j \geq \max\{y, x\} \geq m/(m+1)C^{OPT}$ for any $j \neq i$. The total load of all machines is at least

$$\begin{aligned}\sum_{j \neq i} L_j + L_i &> ((m-1)m/(m+1) + 2m/(m+1))C^{OPT} \\ &= mC^{OPT},\end{aligned}$$

which is a not true since $C^{OPT}$ is at least the average of total loads.

**Case 2** Machine $i$ uses the policy $L$, *i.e.*, $i \in \Psi$. We get that $x \geq y \geq m/(m+1)C^{OPT}$. On the other hand, $L_j \geq x$ for any $j \neq i$ due to the equilibrium. Again, we have the total load is larger than $mC^{OPT}$, which is a contradiction. □

In the following, we show the price of anarchy will be smaller than 4/3 as indicated in Theorem 3. Moreover, we will provide lower bounds for general $m$ machines in the following.

**Theorem 4** *The price of anarchy of the* $(S, L)$*-game for two machines is at most* $9/7$.

***Proof*** If the machine profile of the final equilibrium is $(L, L)$, then it is the same as the LPT algorithm (Graham 1969), which is of $(4/3 - 1/(3m))$-approximation. Thus the $(S, L)$-game has a PoA of $7/6$ for $m = 2$.

Without loss of generality, suppose that job $j$ with the longest completion time is processed on machine 1, whose processing time is $y$. Let $x = L_1 - y$. The load

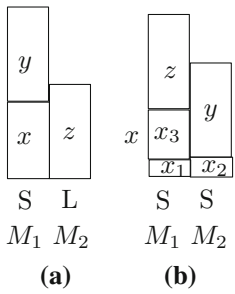


**Fig. 3** Job configuration in case 1

of machine 2 is denoted by $L_2 = z$. Further, we can assume that $C^{OPT} = 1$ and $x + y + z = 2$, otherwise we can add tiny jobs to the instance such that the assumption holds. If the machine profile in the equilibrium is $(S, S)$ or $(L, L)$, the added tiny jobs may increase the makespan. If the machine profile is $(S, L)$, the added tiny jobs will increase the load in the machine with policy S, no matter whether a job in this machine will move to the machine with policy L, the makespan is not decreased. Hence we proved that the price of anarchy is not decreased with this modification of the original instance.

Similar to the general case, we suppose that $y > 4/7$, otherwise, from the Inequality (1), the PoA is at most $9/7$.

**Case 1** The machine profile of the equilibrium is $(S, L)$. First, let us consider the case that the machine that job $j$ is located selects policy S (see Fig. 3a for an illustration of job configuration). We assume that $L_1 = x + y > 9/7$, otherwise it is done. Then $L_2 = z < 5/7$, and this machine only consists of a single job. Let $\varepsilon, \delta > 0$ such that $z = 5/7 - \varepsilon$ and $y = 4/7 + \delta$. Then $x = 2 - y - z = 5/7 + \varepsilon - \delta$.

Once we change the profile to $(S, S)$ (see Fig. 3b for a job equilibrium). Hence we assume that $x_1, x_3$, and $z$ assign to one machine, while $x_2$ and $y$ assign to one machine, where $x_1 + x_2 + x_3 = x$ and $x_1 \leq x_2$ and $x_3$ is the longest jobs among in $M_2$ in Fig. 3a. Notice that in Fig. 3b, job $z$ and $y$ cannot be assigned together, otherwise, $x_3 \geq y$ and then it is a contradiction with $C^{OPT} = 1$.

Suppose that $x_2 \leq 1/7 - \delta - \varepsilon$, otherwise $x_2 + y > z$, and then the profile $(S, L)$ is not stable and it is a contradiction. Hence, $x_1 \leq x_2 \leq 1/7 - \delta - \varepsilon$ and $x_3 \geq 3/7 + \delta + 3\varepsilon$. Note that $x_3$ consists of only one job. Now we get a contradiction that $C^{OPT}$ is 1 and hence $x + y \leq 9/7$.

One can give the analogous proof for job $j$ that is located at the machine with policy L. In this case, $M_1$ uses policy L, and then $x \geq y > 4/7$. Again, suppose that $z = 5/7 - \varepsilon$. We know that no job in $M_2$ has processing time larger than $3/7$, otherwise $C^{OPT} > 1$. If we change the policy of $M_2$ to L, then the two jobs in $M_1$ will be assigned to two machines, and the load in both machines are at least $11/14$, which implies that $M_2$ is benefit by this changing. Hence it is a contradiction with the fact that $(L, S)$ is a Nash equilibrium.

**Case 2** The machine profile of the equilibrium is $(S, S)$. See Fig. 4 for the configuration of the profile $(S, S)$, where $z_2$ is the second largest job. Again, we assume that $5/7 >$

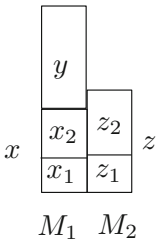


**Fig. 4** Job configuration in the profile $(S, S)$

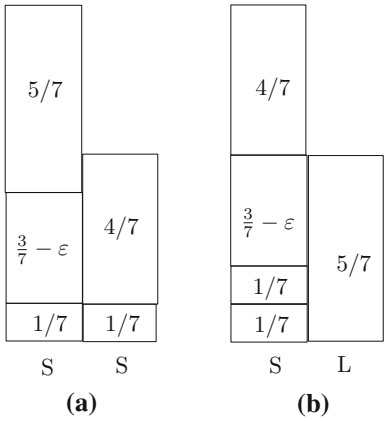


**Fig. 5** Lower bound for the price of anarchy for two machines

$z > y > 4/7$, otherwise, this case is done, and the inequality $z > y$ holds since $M_2$ does not change its policy to L. Since $x + y > 9/7$, we have $x > 4/7$.

Notice that the second longest job $z_2$ must be located in machine $M_2$ due to $(S, S)$ policy. If $z_2 \leq 4/7$, then it will be beneficial if we change the profile to $(S, L)$, since the maximum gap between these two machines is at most the second largest job and then the minimum load of a machine is at least $5/7$ and thus larger than $z$. Now let us consider the case that $z_2 > 4/7$. We have $z_1 < 1/7$, which implies $x_2 > 3/7$ since $x_1 + x_2 = x > 4/7$. Note that $x_2$ consists of only one job and it is a contradiction with $C^{OPT} = 1$. □

**Theorem 5** *The price of anarchy of the* $(S, L)$*-game is at least* $2m/(m+1)$ *for* $m \geq 3$ *and at least* $9/7$ *for* $m = 2$.

***Proof*** For any given sufficiently small $\varepsilon > 0$, we prove that the lower bound for $m \geq 3$ is at least $(2m - \varepsilon)/(m + 1)$, and for $m = 2$ the lower bound is at least $9/7 - \varepsilon$.

We first consider an instance when $m = 2$. There are five jobs with processing times $1/7, 1/7, 3/7 - \varepsilon, 4/7, 5/7$, respectively. Figure 5a is a Nash equilibrium, where the profile of the two machines is $(S, S)$. To see this, note that no job has the incentive to move. Meanwhile, Fig. 5b shows that neither machine has the incentive to change his policy to L.

Clearly, the optimal makespan is 1, while the equilibrium has a length of $9/7 - \varepsilon$. Thus $PoA \geq 9/7$ for $m = 2$.

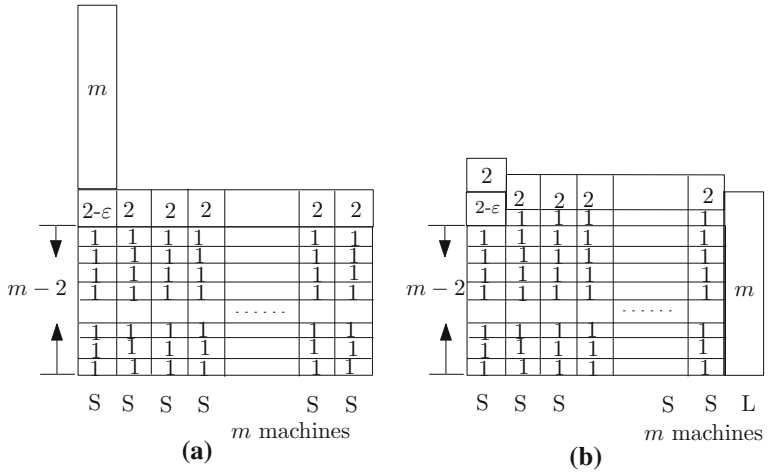


**Fig. 6** Lower bound for the price of anarchy for $m \geq 3$ machines

Now we turn to the general case that $m \geq 3$. In the instance, we are given $m(m-2)$ jobs of processing time 1, $m-1$ jobs of processing time 2, one job of processing time $2-\varepsilon$ and one job of processing time $m$. The machine profile is $(S, S, \ldots, S)$, and the assignment of jobs in Fig. 6a is a Nash equilibrium. To see this, let us consider if one machine (not the first one) changes his policy to L, the unique equilibrium of profile $(S, S, \ldots, S, L)$ is illustrated in Fig. 6b. The load of the machine with policy L is $m$, showing that no improvement at all on the workload, compared with Fig. 6a. Therefore, the makespan is $2m-\varepsilon$, while the optimal makespan is $m+1$. The $PoA \geq 2m/(m+1)$. □

### 3.3 The (*L*, *M*)-game

We consider the $(L, M)$-game, where machines can use the policy LongestFirst or Makespan.

**Theorem 6** *The price of anarchy of the* $(L, M)$*-game is* $2m/(m+1)$ *for* $m \geq 3$.

***Proof*** To prove the lower bound, we are given $(m-1)(m-2)$ jobs of processing times 1, $(m-1)$ jobs of processing time 2, one job of processing time $m-\epsilon$, and one job of processing time $m$, and the profile of machine policies is $(M, M, \ldots, M)$. All the machines use the policy M. The job configuration in Fig. 7a is an equilibrium. This is because all the jobs cannot reduce their completion times if only one of them is moving. On the other hand, if one machine change to L, Fig. 7b is an equilibrium, in which the load of the machine with policy L is $m$. Hence, Fig. 7a is a Nash equilibrium for the $(L, M)$-game, which indicates that the PoA is at least $2m/(m+1)$ for $m \geq 3$.

The proof for the upper bound is shown below. Again let us consider the job $J'$ with processing time $y$ that determines the makespan locates on the machine $i$. Let $x = L_i - y$ be the load on machine $i$ without counting $J'$.

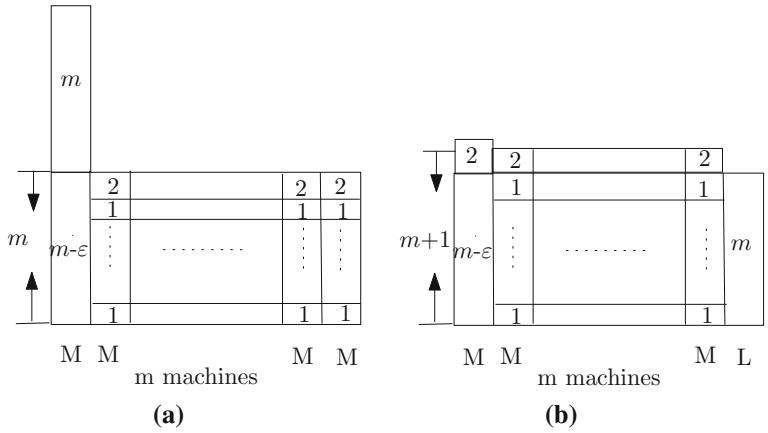


**Fig. 7** Job configuration in the profile $(L, M)$-game

Each machine with policy M has load at least $y$, otherwise it will increase its load by changing to policy L. On the other hand, we know the load in machines with policy L is at least $y$. Similar as Theorem 3, if $y > m/(m+1)C^{OPT}$ and $x + y > 2m/(m+1)C^{OPT}$, where $C^{OPT}$ is the optimal social solution, we will get a contradiction with total optimal load is within $mC^{OPT}$. Then we have either $y \leq m/(m+1)C^{OPT}$ or $x + y \leq 2m/(m+1)C^{OPT}$, which therefore the PoA is at most $2m/(m+1)$. □

From the construction of the lower bound for the $(L, M)$-game, we know that the result of Theorem 3.3 does not valid for $m = 2$. In the following, we obtain a new result when $m = 2$.

**Theorem 7** *The price of anarchy of the* $(L, M)$*-game is* $7/6$ *for* $m = 2$.

***Proof*** To get the lower bound, we consider 2 machines with the profile $(L, L)$, and 5 jobs with processing times $1/2, 1/2, 1/3, 1/3, 1/3$, respectively. Figure 8a gives a Nash equilibrium, which indicates the lower bound of $7/6$. If one machine in Fig. 8a replaces his policy to M, one stable configuration is illustrated in Fig. 8b and the machine with policy M does not increase the load compared to that load in Fig. 8a.

We prove the upper bound by counterproof as below. Clearly, if the profile is $(L, L)$, then it acts the same as the LPT algorithm, which is $(4/3 - 1/(3m))$-approximated and it is 7/6-approximated if $m = 2$ (Graham 1969), and then $PoA \leq 7/6$.

Let us consider the machine profile of the final equilibrium is $(L, M)$. Suppose the job that makes the makespan with processing time $y$ is located on the machine $M_1$, a total load of other jobs on this machine is $x$, and the load of the other machine $M_2$ is $z$. Again, we suppose that $C^{OPT} = 1$, $y > 1/3$, and $x + y > 7/6$. Suppose there are some jobs with processing time no greater than $1/3$, otherwise, the optimal solution cannot be 1.

If the machine $M_2$ uses the policy M and the machine $M_1$ uses policy L, which will give a contradiction. Since, if we change the policy of $M_2$ to L, then the minimum

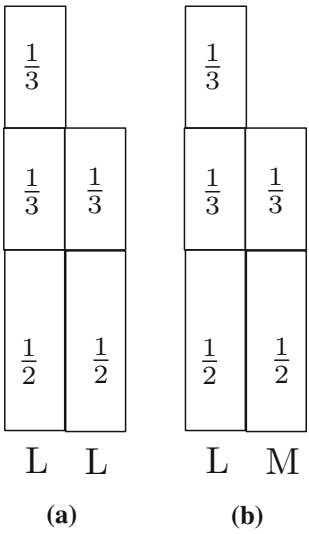


**Fig. 8** Job configuration in the profile $(L, M)$-game when $m = 2$

load among the machines with profile $(L, L)$ is at least $5/6$ which is larger than $z$ since some job is smaller than $1/3$, thus it benefits for $M_2$ changing its policy.

If the machine $M_2$ uses the policy L and the machine $M_1$ uses the policy M, we know that $5/6 > z > y > 1/3$. Notice that all the jobs in $M_1$ must be larger than $1/3$, otherwise, any job with processing time no more than 1/3 will move to $M_2$ such that its completion time is reduced. Now get that all the jobs in the system are larger than 1/3, which conflicts with $C^{OPT} = 1$.

The final case we need to consider is the profile of the final equilibrium is $(M, M)$. Again $5/6 > z > y > 1/3$, all the jobs in $M_1$ consists of jobs larger than 1/3. If $M_2$ changes to L, then, denote $L'_i$ to be the load after changing policy, $i = 1, 2$. We have $L'_2 < z < 5/6$, which indicates all the jobs in $L'_1$ is larger than 1/3. Since $M_2$ taking the policy L after changing, all the jobs in this machine is no less than the jobs in $L'_1$, which indicates that all the jobs in the system are larger than 1/3. Again it conflicts with $C^{OPT} = 1$. □

## 4 The PoA for maximizing the minimum load

In this section, we study the game with the social objective of maximizing the minimum load among machines. The change of social objective does not affect the existence of equilibria. However, the price of anarchy might be very different.

### 4.1 The $(S, L)$-game

**Theorem 8** *The lower bound of the price of anarchy of the $(S, L)$-game is at least $2 - 2/(m - 1)$, and the upper bound is at most $2 - 1/m$, where $m$ is the number of machines.*

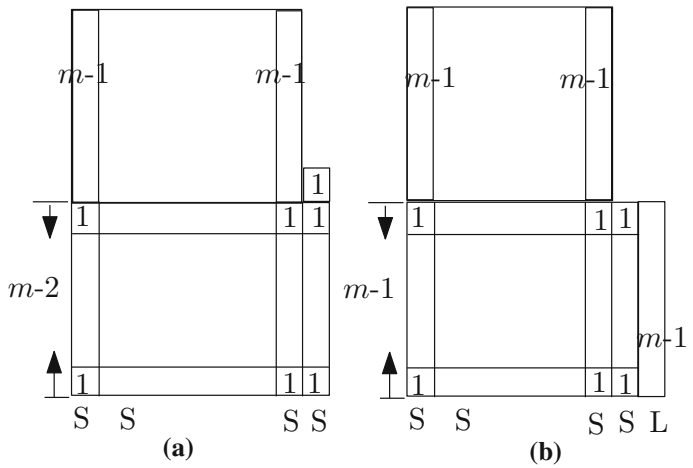


**Fig. 9** Illustration of the lower bound for the $(S, L)$-game in the Max-Min model

***Proof*** The job instance is $m(m-2)+1$ jobs of processing time 1 and $m-1$ jobs of processing time $m-1$. Since all the machines use the policy S, we know that a job configuration in Fig. 9a is a job equilibrium. Figure 9b shows that no machine has the incentive to change its policy to L. From this instance, $C^{NE} = m-1$ and $C^{OPT} = 2m-4$, therefore we obtain that $PoA \geq 2 - 2/(m-1)$.

The upper bound is shown below. Denote $L_j$ to be the load of machine $j$. Suppose that the machine $i$ has the minimum load. Thus $C^{NE} = L_i$. Suppose that the machine $k$ reaches the makespan and let $y = L_k - L_i$. Denote by $J'$ the latest job on the machine $k$, whose processing time is $z$, then $z \geq y$, otherwise job $J'$ would move to the machine $i$. Similar as Theorem 3, we know $L_i \geq \max\{z, L_k - z\}$. Thus, for any other machine $j \neq i$, we obtain that

$$L_j \geq L_i \geq \max\{z, L_k - z\} \geq y,$$

which implies that $y \leq L_i$. On the other hand,

$$\begin{aligned} mC^{OPT} &\leq mL_i + \sum_{j \neq i}(L_j - L_i) \\ &\leq mL_i + (m-1)y \\ &\leq (2m-1)L_i = (2m-1)C^{NE}, \end{aligned}$$

which gives $PoA = C^{OPT}/C^{NE} \leq 2 - 1/m$. □

### 4.2 The (*S*, *M*)-game

**Lemma 9** *The price of anarchy of the* $(S, M)$*-game in the Max-Min model is at least* $m$.

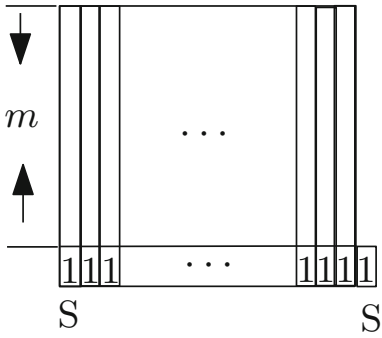


**Fig. 10** Illustration of the lower bound for the $(S, M)$-game in the Max-min model

***Proof*** There are total $m$ jobs of processing time 1 and $m-1$ jobs of processing times $m$. All the machines use the policy S. From Fig. 10, no machine has incentive to change its policy to M. In this equilibrium, $C^{NE} = 1$, while $C^{OPT} = m$. □

**Theorem 10** *The price of anarchy of the* $(S, M)$*-game in the Max-Min model is at most* $m$.

***Proof*** Let $j$ be the machine with the minimum load, and suppose its load is 1. Hence $C^{NE} = 1$. Denote $y_k$ to be the size of the latest job in the machine $k$. Hence the load in machine $k \neq j$ is at most $1 + y_k$, otherwise that job with size $y_k$ will switch to the machine $j$.

Suppose that we keep the last job of all that $m-1$ machines other than $j$, and move the other jobs to the machine $j$. In this case, machine $j$ has a load at most of $m$, and the other machines contain a single job. It is easy to check that the load of machine $j$ is an upper bound of the optimal solution. As a consequence, $C^{OPT} \leq m$. □

### 4.3 The (*L*, *M*)-game

If all the machines use policy L, Deuermeyer et al. (1982) have shown the upper bound is at most of $4/3 - 1/3m$. Epstein et al. (2009) have proved that if all the machines use policy M, the price of anarchy is bounded by 1.691 and 1.7. In this section, we will prove that the price of anarchy of the $(L, M)$-game is also between 1.691 and 1.7. The technique for the upper bound proof is based on Epstein et al. (2009), some analysis in that paper will be adopted as a black-box.

**Theorem 11** *The lower bound of the price of anarchy for the* $(L, M)$*-game in the Max-Min model approaches to* 1.691 *when* $m$ *is sufficiently large.*

***Proof*** Let $t_1 = 1$ and $t_{i+1} = t_i(t_i + 1)$ for $i \geq 1$. Let $m = t_k$ for an integer $k$. All the machines adopt the Makespan policy. Let $\varepsilon > 0$ be a sufficiently small number. The job configuration consists of $k-1$ classes. The first class has $m/2$ machines, with each machine has 1 job of processing time 1 and 1 job of processing time $1 - \varepsilon$. For class $2 \leq i \leq k-2$, there are $m/(t_i + 1)$ machines, with each machine has $t_i + 1$ jobs of processing time $1/t_i$. The last class consists of $m/(t_{k-1} + 1)$ machines, and each

machine has $t_{k-1} - 2$ jobs of processing time $1/t_{k-1}$ and one job of processing time $3/t_{k-1}$. Denote $L_i$ to be the load of a machine in each class $i$. Thus in this configuration, we have $L_1 = 2 - \varepsilon$, $L_i = 1 + 1/t_i$ for $2 \le i \le k - 1$.

First, this configuration is a job equilibrium. Any job in the smaller class do not move the machines in a larger class, $L_i < L_j$ if $i > j$. Moreover, the jobs in class $i$ do not move to the class $j$ with $i \le j$, since the load in machine of class $i$ will be 1 by removing one job, while $L_j > 1$ for any $j$.

At second, no machine has an incentive to change its policy to L.

**Case 1** One of the machines $M_m$ from the last class changes its policy. We know the last class has $t_{k-1}$ machines. The load in this class before changing is $1 + 1/t_{k-1}$. After changing, the machine $M_m$ has load $1 + 1/t_{k-1}$, and the other load of this class is $1 + 2/t_{k-1}$. This is because one job of processing time 1 from the first class will move to the machine $M_m$, follows which a job of $1/t_{k-1}$ on machine $M_m$ moves to different machines in the $(k - 1)$th class, and then the job of $3/t_{k-1}$ on machine $M_m$ move to the least loaded machine on the first class.

**Case 2** One of the machine except the last class changes its policy, suppose the machine to be $M_j$ in class $j$ for $1 \le j \le k - 2$. The number of jobs in $M_j$ is $t_j + 1$, the next class has machines $m/(t_j + 1) \ge t_j$ for $1 \le j \le k - 2$. Thus it is an equilibrium if $t_j$ jobs would move to the next class such that some machine increases load at most $1/t_j$, thus whose load is $1 + 1/t_{j+1} + 1/t_j$, which will finally converge to the situation that $M_j = 1 + 1/t_j$, since $1 + 2/t_j > 1 + 1/t_{j+1} + 1/t_j$. As a consequence, the machine $M_j$ has no incentive to change its policy.

From the above analysis, $C^{NE} = 1 + 1/t_{k-1}$, while we can prove that $C^{OPT} = \sum_{i=1}^{k-2} t_i - \varepsilon$, since in the optimal solution, each machine has a set of jobs of distinct sizes 1 (or $1-\varepsilon$) , $1/2, \ldots, 1/t_{k-2}$. Thus, let $m = t_k$ and $k$ be a sufficient large integer, $\varepsilon$ approaches to zero, we get that $PoA \ge C^{OPT}/C^{NE} \to 1.691$. □

In the following, we show that the upper bound of PoA for $(L, M)$-game is at most 1.7,

**Theorem 12** *The upper bound for the price of anarchy of the* $(L, M)$*-game in the Max-Min model is at most* 1.7.

***Proof*** Epstein et al. (2009) proved that the PoA for the $(M, M)$-game is at most of 1.7. Hence, we only consider the final equilibrium with policies M and L, or only with L. Denote $C^{NE}$ to be the minimum machine load in the final equilibrium and $C^{OPT}$ to be the optimal value. Without loss of generality, we assume that $C^{NE} = 1$. If the makespan of the equilibrium is at most 1.7, we obtain that $C^{OPT} \le 1.7$ and then the theorem is done. In the following, we assume that the makespan is generated on machine $M_j$ which is larger than 1.7.

**Case 1** The machine $M_j$ uses the policy L. The last job in $M_j$ are at least 0.7, and then all the jobs in $M_j$ are no less than 0.7. Thus it has only two jobs in this machine. We also obtain that all the jobs in machines with L is no less than 0.7.

**Case 2** The machine $M_j$ uses the policy M. The last job in $M_j$ is at least 0.7, all the jobs in machines with policy L is at least 0.7.

Hence, we obtain that all jobs in machines with L are no less than 0.7, and there are at most two jobs in such machines.

We denote the machine with load 1 in the final equilibrium NE by $P$. All other machines are called *tall machines*. Let $C^{OPT}$ be the optimal solution. Obviously, $C^{OPT} \geq 1$. We modify the instance as Epstein et al. (2009) one by one on NE, which does not increase $C^{OPT}$ and does not violate the equilibrium conditions for NE and does not decrease $C^{NE}$.

- Step 1: Remove the machine and all the jobs in this machine if there is a single job in a machine.
- Step 2: Enlarge the jobs to be 1 if a tall machine in NE which has two jobs.
- Step 3: Given jobs of sizes $p_1 \leq p_2 \leq \cdots \leq p_t$ assigned to a machine $Q \neq P$ in NE, then increase $p_t$ such that $p_2 + \cdots + p_t = 1$.
- Step 4: If a machine $Q \neq P$ in NE which has $t \geq 3$ jobs assigned to it. Replace the largest job with two jobs of its half size such that the largest jobs are no more twice larger than the smallest job.
- Step 5: Partition jobs such that machine P contains only tiny jobs, that is, jobs of infinitesimal size.

The above steps are the same as reference Epstein et al. (2009), we can also check that the actions do not affect by the machines with policy L. Step 1 ensures that there is no single job with size larger than 1. Step 2 guarantees the machines with policy L have exactly two jobs whose size now is 1. Step 3 does not violate the NE since $p_t$ is not larger than 1 and it does not move to the machines with policy L. Step 4 and Step 5 only decrease the sizes of the corresponding jobs, they will not change the allocation of the machine with policy L. From Step 3 and Step 4 we know there is no job of size between $[2/3, 1)$.

By the above modification, we adopt the same weight function $w(x)$ as Epstein et al. (2009) on sizes of jobs $x$.

$$w(x) = \begin{cases} \frac{1}{2} & \text{for } x = 1, \\ \frac{x}{2-x} & \text{for } x \in (1/2, 2/3), \\ \frac{x}{x+1} & \text{for } x \in (0, 1/2]. \end{cases}$$

First, machine P has a weight of less than 1. The total weight (by $w$) of jobs assigned to a machine $Q \neq P$ in NE is at most 1 since all the machines with policy L have only two jobs and their total weight is exactly 1. From Epstein et al. (2009), we know that machines use policy M have weight at most of 1. Thus, any optimal solution must have one machine with weight strictly less than 1. By the proof of Lemma 7 in Epstein et al. (2009), we know the total size of jobs with weight less than 1 is at most of 1.7. Hence, our proof is done. □

## 5 Concluding remarks

In this paper, we have addressed the price of anarchy in two-stage identical parallel machine scheduling games. In these games, both the machines and the jobs are selfish

players. The prices of anarchy were explored with respect to two social objectives, the minimum of the makespan and the maximum of the minimum machine completion time. We provided nearly tight bounds for the $(S, L)$-game, the $(S, M)$-game, and the $(L, M)$-game, respectively.

Many directions for future work arise from these two-stage games. It is interesting to consider different individual value functions of the agents or different social objectives. In particular, one may consider the case that each machine attempts to minimize his load because each machine might get more profit in the long run since agents prefer a machine with the lightest load. One can extend our work to other measures of efficiencies, such as the strong price of anarchy or price of stability. Another direction might be the extension work on uniformly related machine scheduling, or unrelated machine scheduling.

**Acknowledgements** The authors thank anonymous referees for helpful comments and suggestions to improve the presentation of this paper.

## References

Ashlagi I, Tennenholtz M, Zohar A (2010) Competing schedulers. In: Proceedings of the 24th AAAI conference on artificial intelligence (AAAI), pp. 691–696

Azar Y, Jain K, Mirrokni V (2008) (Almost) optimal coordination mechanisms for unrelated machine scheduling. In: Proceedings of the 19th annual ACM-SIAM symposium on Discrete algorithms (SODA), pp 323–332

Caragiannis I (2009) Efficient coordination mechanisms for unrelated machine scheduling. In: Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 815–824

Chen X, Epstein L, Kleiman E, van Stee R (2013) Maximizing the minimum load: the cost of selfishness. Theoret Comput Sci 482:9–19

Chen X, Hu X, Ma W, Wang C (2012) Efficiency of dual equilibria in selfish task allocation to selfish machines. In: Proceedings of the 6th international conference on combinatorial optimization and applications (COCOA), pp. 312–323

Christodoulou G, Koutsoupias E, Nanavati A (2004) Coordination mechanisms. In: Proceedings of the 31st international colloquium on automata, languages, and programming (ICALP), pp 45–56

Cohen J, Dürr C, Nguyen Kim T (2011) Non-clairvoyant scheduling games. Theory Comput Syst 1–21

Cole R, Correa JR, Gkatzelis V, Mirrokni V, Olver N (2011) Inner product spaces for minsum coordination mechanisms. In: Proceedings of the 43rd annual ACM symposium on theory of computing (STOC), pp 539–548

Deuermeyer BL, Friesen DK, Langston MA (1982) Scheduling to maximize the minimum processor finish time in a multiprocessor system. SIAM J Algebraic Discrete Methods 3(2):190–196

Dobson G (1984) Scheduling independent tasks on uniform processors. SIAM J Comput 13:705–716

Dubey P (1986) Inefficiency of Nash equilibria. Math Oper Res 11(1):1–8

Epstein L, Kleiman E, Stee R (2009) Maximizing the minimum load: the cost of selfishness. In: Proceedings of the 5th international workshop on internet and network economics (WINE), pp 232–243

Epstein L, Kleiman E, Stee R (2014) The cost of selfishness for maximizing the minimum load on uniformly related machines. J Comb Optim 27(4):767–777

Epstein L, Stee R (2012) The price of anarchy on uniformly related machines revisited. Inf Comput 212:37–54

Fabrikant A, Luthra A, Maneva E, Papadimitriou C, Shenker S (2003) On a network creation game. In: ACM Symposium on principles of distributed computing (PODC), pp 347–351

Feldman M, Immorlica N, Lucier B, Roughgarden T, Syrgkanis V (2016) The price of anarchy in large games. In: Proceedings of the 48th annual ACM symposium on theory of computing (STOC), pp 963–976

Finn G, Horowitz E (1979) A linear time approximation algorithm for multiprocessor scheduling. BIT Numer Math 19(3):312–320
Friesen DK (1987) Tighter bounds for LPT scheduling on uniform processors. SIAM J Comput 16:554–560
Graham RL (1966) Bounds for certain multiprocessing anomalies. Bell System Tech J 45:1563–1581
Graham RL (1969) Bounds on multiprocessing timing anomalies. SIAM J Appl Math 17:263–269
Heydenreich B, Müller R, Uetz M (2007) Games and mechanism design in machine scheduling—an introduction. Prod Oper Manag 16(4):437–454
Hoeksma R, Uetz M (2011) The price of anarchy for minsum related machine scheduling. In: Proceedings of the 9th international conference on approximation and online algorithms (WAOA), pp 261–273
Immorlica N, Li LE, Mirrokni VS, Schulz AS (2009) Coordination mechanisms for selfish scheduling. Theor Comput Sci 410:1589–1598
Koutsoupias E, Papadimitriou C (1999) Worst-case equilibria. In: Proceedings of the 16th symposium on theoretical aspects of computer science (STACS), pp 404–413
Lin L, Tan Z (2014) Inefficiency of Nash equilibrium for scheduling games with constrained jobs: a parametric analysis. Theor Comput Sci 521:123–134
Lucier B, Borodin A (2010) Price of anarchy for greedy auctions. In: Proceedings of the 21st annual ACM-SIAM symposium on discrete algorithms (SODA), pp 537–553
Nisan N, Roughgarden T, Tardos E, Vazirani VV (2007) Algorithmic game theory. Cambridge University Press, Cambridge
Roughgarden T, Tardos E (2004) Bounding the inefficiency of equilibria in nonatomic congestion games. Games Econ Behav 47(2):389–403
Roughgarden T, Tardos E (2002) How bad is selfish routing? J ACM 49(2):236–259
Tan Z, Wan L, Zhang Q, Ren W (2012) Inefficiency of equilibria for the machine covering game on uniform machines. Acta Inform 49(6):361–379
Vetta AR (2002) Nash equilibria in competitive societies with applications to facility location, traffic routing and auctions. In: Symposium on the foundations of computer science (FOCS), pp 416–425