ENGINEERING Information Technology & Electronic Engineering
https://www.academax.com/EITEE; https://www.engineering.org.cn
https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=11344065
E-mail: eng.itee@zju.edu.cn

Engineering Journals
Available on IEEE Xplore®

Research Article
https://doi.org/10.1631/ENG.ITEE.2026.0111

Exact price of anarchy for a two-uniform-machine scheduling game under parallel processing

Yong WU, Qi WEI✉

College of Management, Ningbo University of Finance and Economics, Ningbo 315175, China

Abstract: This study analyzes the price of anarchy (PoA) for a scheduling game on two hierarchical uniform machines under the parallel processing policy, a coordination mechanism that combines the well-studied ShortestFirst and makespan policies. This scheduling game corresponds to the round-robin scheduling mechanism widely adopted in time-sharing operating systems and network packet processors. This study focuses on makespan minimization. We derive the PoA as a function of the ratio of the speeds of the two machines. We demonstrate that all derived bounds are tight for any ratio, thereby characterizing the inefficiency of Nash equilibria in scheduling games on two uniform machines under parallel processing.

Key words: Scheduling game; Hierarchy; Parallel processing; Price of anarchy (PoA); Nash equilibrium

1 Introduction

This study investigates a distributed scheduling problem on two hierarchical uniform machines, where jobs are classified into two hierarchical classes. Conventional scheduling research typically assumes a centralized controller that collects information from all users and coordinates their access to shared resources. However, such centralized coordination is often infeasible in practical distributed scheduling scenarios, such as network routing. In distributed scheduling environments (formally defined as scheduling games), decisions are made by selfish jobs (i.e., players in scheduling games) that are only interested in achieving their objectives without any concern for the global optimal outcome for the game. The objective of each job is to minimize its completion time. Jobs have two available strategy types: pure and mixed strategies. A pure strategy implies that a selfish job must select one specific machine; a mixed strategy stipulates a probability distribution over the set of pure strategies. A Nash equilibrium is a schedule where no job has a unilateral incentive to switch to another machine. Throughout this study, we only consider pure strategies and pure Nash equilibria. Beyond individual job objectives, the system pursues a global social objective, namely, makespan minimization, which is the most widely adopted criterion in the scheduling literature. This objective arises from practical demands for guaranteed quality of service and equitable resource allocation, which is defined as minimizing the maximum completion time across all machines or as the makespan (Finn and Horowitz, 1979). Many studies have addressed performance degradation caused by a lack of coordination in such selfish scheduling games.

The scheduling game investigated in this study is motivated by practical engineering scenarios in which heterogeneous resources serve users with different privileges. In cloud computing platforms, virtual machines with different processing speeds are shared among numerous independent users, each submitting computational tasks and aiming to minimize their job completion time (Chen C and Xu, 2020). Similarly, in central processing unit (CPU)–graphics processing unit (GPU) heterogeneous computing systems, certain tasks (e.g., video processing) are processed significantly faster on GPUs, whereas other tasks remain more efficient on CPUs; the operating system's scheduler must allocate these tasks without full centralized control (Chen C and Xu, 2020). The parallel processing policy analyzed in this study corresponds to the round-robin scheduler commonly implemented in time-sharing operating systems and network packet processors, where each active job or flow receives a fair share of the processing resource (Yu et al., 2010; Cohen et al., 2011).

Mechanism design provides a powerful framework for mitigating equilibrium inefficiency by guiding selfish job behaviors toward desirable system outcomes. Representative approaches

✉ Qi WEI, weiqi@nbufe.edu.cn
Yong WU, https://orcid.org/0000-0001-8515-708X
Qi WEI, https://orcid.org/0000-0001-9852-5902
CLC number: TP301.6; O223

Received: Apr. 19, 2026; Revision accepted: Aug. 19, 2026;
Crosschecked: Sept. 14, 2026; Published online: Oct. 6, 2026

© The Authors 2026. Published by Zhejiang University Press Co., Ltd. This is an open access article distributed under the terms of the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0/)

include congestion tolls (Cole et al., 2006), the Stackelberg strategy (Roughgarden, 2001), and coordination mechanisms (Christodoulou et al., 2004; Azar et al., 2015; Kress et al., 2018; Chen C and Xu, 2020). First formalized by Christodoulou et al. (2004), coordination mechanisms constitute a core research direction in scheduling games. A coordination mechanism defines a local scheduling rule for each machine that determines the processing order of assigned jobs (Immorlica et al., 2009). Its core design goal is to guarantee the existence of Nash equilibria for the induced game. The price of anarchy (PoA), introduced by Koutsoupias and Papadimitriou (1999), has become the standard metric for measuring efficiency loss from selfish job behaviors. The PoA of a game equals the ratio between the social objective value of the worst-case Nash equilibrium and the optimal social objective value.

There are several classical and well-studied policies such as the ShortestFirst, LongestFirst, randomized policy, and makespan policy. Under ShortestFirst/LongestFirst, each machine processes jobs in nondecreasing/nonincreasing order of their processing times (Christodoulou et al., 2004; Immorlica et al., 2009; Muñoz and Linfati, 2024). Under the randomized policy, each machine processes jobs randomly (Czumaj and Vöcking, 2007; Immorlica et al., 2009; He and Tan, 2026). The makespan policy equalizes the completion time of all jobs assigned to the same machine (Czumaj and Vöcking, 2007; Lin and Tan, 2014). Cohen et al. (2011) proposed the EQUI policy, also known as the parallel processing policy (Yu et al., 2010). Under this policy, a machine splits its timeline into equal-length time slots and serves all unfinished jobs in a round-robin manner. This study focuses solely on the parallel processing coordination mechanism.

Bar-Noy et al. (2001) originally proposed hierarchical scheduling. Service hierarchy models differentiated privilege-based resource access, a ubiquitous practice across service industries, computer systems, telecommunication networks, and hierarchical databases (Bar-Noy et al., 2001; Chen QQ and Tan, 2021; Lin and Tan, 2025). We adopt the standard three-field scheduling notation: P denotes identical parallel machines, Q denotes uniform-speed machines, and GoS indicates grade-of-service hierarchical access constraints. $Pm|\text{GoS}|C_{\max}/Qm|\text{GoS}|C_{\max}$ represents the m-machine hierarchical scheduling game for identical/uniform machines with makespan minimization $C_{\max}$.

This study investigates scheduling games for makespan minimization on two hierarchical machines under parallel processing. If all jobs belong to a single service class, the model degenerates to the standard unconstrained uniform machine scheduling problem $Qm||C_{\max}$. Guan and Li (2013) first investigated the selfish scheduling game on hierarchical machines under the makespan and LG-LPT policies. Under the LG-LPT policy, each machine processes jobs in descending order of their classes, whereas jobs in the same class are processed in nonincreasing order of their processing times. The researchers derived the PoA for the makespan minimization problem on m identical parallel machines. Under the parallel processing policy, Yu et al. (2010) presented upper and lower bounds on PoAs for various scheduling problems. Wan et al. (2013) demonstrated that the exact PoAs for $Pm||C_{\max}$ and $Q2||C_{\max}$ are $2-1/m$ and $\min\{\frac{1+s}{s},\ \frac{1+2s}{1+s}\}$, respectively, where $s \geq 1$ denotes the ratio of the speeds of the two machines.

We consider the case involving two machines with different certificates, with jobs belonging to different classes. We obtain the exact PoA as a function of the ratio of the speeds of the two machines. Notably, the existence of jobs of class 1 renders the analysis of PoA difficult. Under the parallel processing policy, the exact PoA for $Q2|\text{GoS}|C_{\max}$ is $f(s)=\min\{1+s,\ \max\{\frac{2+s}{1+s},\ \frac{1+2s}{1+s}\}\}$. If $s=1$, the exact PoA for $P2|\text{GoS}|C_{\max}$ is $3/2$. Parallel processing hybridizes two classic policies: ShortestFirst and makespan. Epstein (2010) and Wu et al. (2015) derived the exact PoA values for hierarchical two-machine scheduling under the pure makespan policy. Under the makespan policy, the completion time of a job is equal to the completion time of the machine on which it is processed. In contrast, under parallel processing, only the largest job on a machine incurs a completion time equal to the machine load, which substantially complicates equilibrium analysis. Moreover, we intend to determine whether the parallel processing policy reduces PoA under the makespan policy. Intuitively, the parallel processing policy should be superior to the makespan policy because it divides the time horizon into tiny time slots of an equal length and processes all currently unfinished jobs in parallel. However, as illustrated in Fig. 1, the PoA curve of parallel processing always lies above or coincides with that of makespan for all $s>0$, implying that parallel processing cannot outperform makespan in terms of equilibrium efficiency loss. Therefore, equal time-slice allocation does not reduce selfish inefficiency for hierarchical scheduling games.

2 Preliminaries

We consider two uniform machines, namely, M_1 and M_2, and a job set $\mathcal{J}$ containing n independent selfish jobs $J_1, J_2, \ldots, J_n$. The speed of M_t is denoted by s_t. Without loss of generality, we assume that $s_1=1$ and $s_2=s>0$. Each selfish job $J_i=(p_i,\ g_i)$ is a player of the game, where p_i and $g_i \in \{1,2\}$ denote the size and class of J_i, respectively. The size p_i is equal to the processing time of job J_i on a unit-speed machine. Moreover, the processing time of this job on a machine of speed s_t is given by p_i/s_t. Machine M_t has a certificate $g(M_t)=t$ $(t=1,2)$, associated with it. Job J_i can be executed on machine M_t only when $g_i \geq g(M_t)$. This constraint forces all class 1 jobs to be scheduled exclusively on M_1. Each class 2

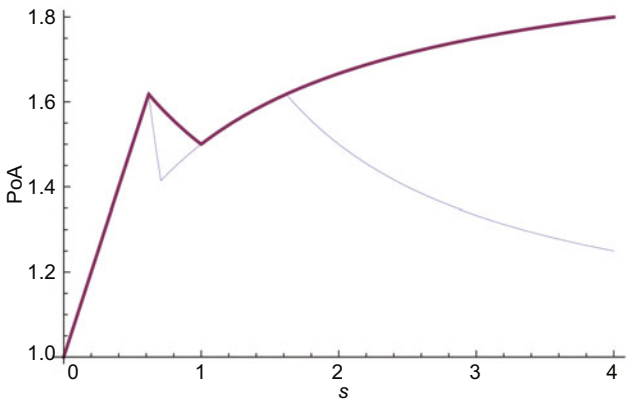


Fig. 1 PoAs for $Q2|\text{GoS}|C_{\max}$ under parallel processing (thick line) and makespan (thin line) policies

job selects a machine to minimize its completion time, and the collective assignment of all jobs defines a schedule. A Nash equilibrium σ denotes a schedule in which no job can lower its completion time by unilaterally switching machines. Yu et al. (2010) proved that the Nash equilibrium always exists under the parallel processing policy. Crucially, the existence of the Nash equilibrium is only related to the coordination mechanism. Thus, the Nash equilibrium of a scheduling game under the parallel processing policy always exists regardless of the social objective used. For the game $Qm|\mathrm{GoS}|C_{\max}$, the parallel processing policy is adopted, and we obtain the following results:

Lemma 1 There exists a Nash equilibrium for $Qm|\mathrm{GoS}|C_{\max}$ under the parallel processing policy (Yu et al., 2010).

For any Nash equilibrium σ, suppose that n_i jobs $J_{1t}, J_{2t}, \cdots, J_{n_i t}$ are assigned to machine M_t, sorted in nondecreasing order of job size $p_{1t} \le p_{2t} \le \cdots \le p_{n_i t}$. Under parallel processing, time slots are fairly shared among the current unfinished jobs on a machine, yielding the following completion time C_{kt} formula for the k^{th} ordered job:

$$C_{kt} = \left((n_i - k) p_{kt} + \sum_{1 \le j \le k} p_{jt} \right) \Big/ s_t, \tag{1}$$

where $C_{1t} \le C_{2t} \le \cdots \le C_{n_i t}$ and $C_{n_i t} = \sum_{1 \le j \le n_i} p_{jt}/s_t$. The load of a machine is defined as the total processing time of all jobs assigned to it. Therefore, we obtain the following remark:

Remark 1 Under the parallel processing policy, the load of a machine is equal to the completion time of the largest job processed on it.

Let L_t be the load of machine M_t, $t = 1, 2$, in any Nash equilibrium σ and $\{a, b\}$ be a permutation of $\{1, 2\}$. Let job J_{Bt} be the largest class 2 job scheduled on machine M_t, $t = a, b$, and $J_{\max} = (p_{\max}, 2)$ be the largest job of all class 2 jobs with $p_{\max}$ denoting the max size. Considering that each class 2 job in any Nash equilibrium σ cannot reduce its completion time by moving to a different machine, we obtain the following remark:

Remark 2 Under the parallel processing policy, any Nash equilibrium σ satisfies $L_b + p_{Ba}/s_b \ge L_a$ if the job $J_{Ba} = (p_{Ba}, 2)$ is the largest assigned to machine M_a.

Throughout subsequent analyses, σ denotes an arbitrary pure Nash equilibrium, and L^σ and L^* denote the social objective values generated by σ and the optimal solution, respectively. In the analysis of the Nash equilibrium σ, let G_{2i} be the set of class 2 jobs assigned to machine M_t, $t = 1, 2$, and let G_1 be the set of class 1 jobs. Let $t(\delta)$ be the total size of the jobs in any job set δ.

Under the parallel processing policy, an optimal schedule of the game may not be a Nash equilibrium; the worst-case Nash equilibrium can yield a strictly larger makespan (i.e., social objective value) than the global optimum, as exemplified by the following five-job two-machine game:

Example 1 There are two identical parallel machines, namely, M_1 and M_2, and a job set $\mathcal{J}$ with five jobs, namely, $J_1 = (3,\ 1)$, $J_2 = J_3 = J_4 = (1,\ 2)$, and $J_5 = (6,\ 2)$. An optimal schedule assigns job J_5 to machine M_2 and the other four jobs to machine M_1. Then, we obtain $L^* = 6$ for the game $P2|\mathrm{GoS}|C_{\max}$. The completion time of job J_2 is $p_2 + p_3 + p_4 + p_2 = 4$ in the optimal schedule. If job J_2 is moved to machine M_2, its completion time is $p_2 + p_2 = 2$. Thus, the optimal schedule is not a Nash equilibrium because job J_2 can reduce its completion time by moving to machine M_2. Next, we consider a schedule σ that assigns jobs J_1 and J_5 to machine M_1 and the other three jobs $(1,\ 2)$ to machine M_2. Note that the class of job J_1 is 1; thus, it must select machine M_1. The completion time of job J_5 in σ is $p_5 + p_1 = 9$. If job J_5 moves to a different machine, its completion time remains 9. The completion time of job J_i, $2 \le i \le 4$, in σ is $p_2 + p_3 + p_4 = 3$. If it moves to a different machine, its completion time remains 3. Thus, schedule σ is a Nash equilibrium, and $L^\sigma = 9$ for game $P2|\mathrm{GoS}|C_{\max}$.

3 Upper bound of PoA

This section derives the upper PoA bound for the selfish scheduling game with a makespan minimization objective. Let $\mathcal{I}$ denote the set of all instances and $\Lambda(I)$ denote the set of all pure Nash equilibrium schedules for the instance $I \in \mathcal{I}$. By definition, we have

$$\mathrm{PoA} = \sup_{I \in \mathcal{I}} \sup_{\sigma \in \Lambda(I)} \frac{L^\sigma}{L^*}. \tag{2}$$

First, we present a lemma generalizing a lower bound on the social objective value L^*. The result is useful throughout this section.

Lemma 2 For the makespan minimization problem on two hierarchical uniform machines, we obtain $L^* \ge \max\left\{ \frac{L_1 + sL_2}{1+s},\ \min\left\{ p_{\max},\ \frac{p_{\max}}{s} \right\} \right\}$.

Proof $L^* \ge \frac{L_1 + sL_2}{1+s}$ because $L_1 + sL_2$ denotes the total size of all jobs. In the optimal solution, one machine must process job $J_{\max} = (p_{\max}, 2)$. Thus, we obtain $L^* \ge \min\{p_{\max},\ \frac{p_{\max}}{s}\}$.

Next, we analyze the upper bound of L^σ/L^*, which helps us determine the exact PoA for the scheduling game $Q2|\mathrm{GoS}|C_{\max}$. We distinguish two cases according to the machine that determines the makespan of the worst-case Nash equilibrium σ. Before that, we first present a lemma to facilitate our analysis of the completion time of certain jobs. Let $\alpha \subsetneqq G_{21}\ (\alpha \ne \varnothing)$ and job $J_{k1} = (p_{k1}, 2)$ (may be J_{B1}) be the largest job in the job set α. The selected job J_{k1} is a maximum-size job in α with the largest global index $k1$ in the nondecreasing ordering of all jobs on M_1 (i.e. the last-ranked maximum-size job in the job set $\alpha \subsetneqq G_{21}$) rather than being an arbitrary maximum-size class 2 job in α. The following lemma establishes a lower bound on the completion time of job J_{k1}:

Lemma 3 In any Nash equilibrium σ, suppose that $J_{k1} = (p_{k1}, 2)$ is the last-ranked maximum-size job in the job set $\alpha \subsetneqq G_{21}$ and $t(G_1) > p_{B1}$. Then, the completion time of job J_{k1} satisfies $C_{k1} \ge t(\alpha) + p_{k1}$.

Proof We obtain $C_{k1} = ((n_1 - k)p_{k1} + \sum_{1 \le j \le k} p_{j1})$. If $k = n_1$, $\sum_{1 \le j \le k} p_{j1} = t(G_{21}) + t(G_1)$; thus, $C_{k1} = t(G_{21}) + t(G_1) > t(\alpha) + p_{k1}$ because $t(G_{21}) \ge t(\alpha)$ and $t(G_1) > p_{B1} \ge p_{k1}$. Conversely, if $k < n_1$, $(n_1 - k)p_{k1} \ge p_{k1}$. Because J_{k1} is the last-ranked maximum-size job in $\alpha \subsetneqq G_{21}$, all jobs of α appear no later than position k, which implies $\sum_{1 \le j \le k} p_{j1} \ge t(\alpha)$. Therefore, $C_{k1} \ge t(\alpha) + p_{k1}$ holds, and we obtain the result.

Lemma 4 If $L^\sigma = L_1 \geq L_2$, then $L^\sigma \leq \min\{1+s,\ \max\{\frac{2+s}{1+s},\ \frac{1+2s}{1+s}\}\}L^*$.

Proof By Lemma 2, we obtain $L^\sigma = L_1 \leq L_1 + sL_2 \leq (1+s)L^*$, which yields

$$\frac{L^\sigma}{L^*} \leq 1+s. \tag{3}$$

Next, we prove that $\frac{L^\sigma}{L^*} \leq \max\{\frac{2+s}{1+s},\ \frac{1+2s}{1+s}\}$.

Let us consider the optimal schedule and the worst-case Nash equilibrium σ of the instance by cases.

Case 1: $L_1 = t(G_1)$.

Here, $L_1 = t(G_1) \geq L_2 = t(G_{22})/s$. Because all jobs scheduled on machine M_1 in schedule σ belong to class 1, the Nash equilibrium σ also serves as the optimal schedule. Therefore, $L^\sigma = L^*$.

Case 2: $L_1 > t(G_1)$.

Here, $L_1 = t(G_1) + t(G_{21})$ and $t(G_{21}) > 0$. There exists at least one job of class 2 being assigned to machine M_1 in σ. Recall that job $J_{B1} = (p_{B1},\ 2)$ is the largest class 2 job processed on machine M_1. Following Remark 1, the completion time of job J_{B1} in schedule σ is confirmed to be L_1 when $t(G_1) \leq p_{B1}$. If $t(G_1) > p_{B1}$, the completion time of job J_{B1} in schedule σ cannot be confirmed to be L_1 because job J_{B1} may not be the largest job on machine M_1. Therefore, we distinguish the following two subcases:

Subcase 2.1: $t(G_1) \leq p_{B1}$.

Following Remark 2, we obtain $sL_2 + p_{B1} \geq sL_1$, demonstrating that

$$(1+s)L^\sigma = L_1 + sL_1 \leq L_1 + sL_2 + p_{B1}. \tag{4}$$

Following Lemma 2, we obtain

$$p_{B1} \leq \max\{sL^*,\ L^*\},\ L_1 + sL_2 \leq (1+s)L^*. \tag{5}$$

Inequalities (4) and (5) yield the following results:

$$(1+s)L^\sigma \leq \max\{(2+s)L^*,\ (1+2s)L^*\}. \tag{6}$$

Subcase 2.2: $t(G_1) > p_{B1}$.

Here, $G_{21} \neq \varnothing$, and job $J_{B1} = (p_{B1},\ 2)$ is the largest class 2 job processed on machine M_1. Let $G_{21} \neq \varnothing$ and $J_{k1} = J_{B1}$ (given two or more jobs of type $(p_{B1},\ 2)$, the job with the largest global index in the nondecreasing ordering of all jobs on M_1 is selected). Lemma 3 demonstrates that $C_{B1} \geq t(G_{21}) + p_{B1}$. Because schedule σ is a Nash equilibrium, we obtain

$$L_2 + \frac{p_{B1}}{s} = \frac{t(G_{22})}{s} + \frac{p_{B1}}{s} \geq C_{B1},$$

$$\frac{t(G_{22})}{s} + \frac{p_{B1}}{s} \geq t(G_{21}) + p_{B1}. \tag{7}$$

Next, we prove the result based on the value of s.

Subcase 2.2.1: for $s \geq 1$, inequality (7) yields

$$t(G_{21}) \leq \frac{t(G_{22})}{s}. \tag{8}$$

If $t(G_1) > \frac{t(G_{21})+t(G_{22})}{s}$, the optimal schedule assigns all jobs separately according to their classes. In other words, we obtain $L^* = t(G_1)$. Along with inequality (8), we obtain

$$t(G_{21}) < s \cdot t(G_1) - t(G_{22}) < s \cdot t(G_1) - s \cdot t(G_{21}),$$

so $t(G_{21}) < \frac{s}{1+s}t(G_1) = \frac{s}{1+s}L^*$. Consequently, we obtain

$$L^\sigma = L_1 = t(G_1) + t(G_{21}) < \frac{1+2s}{1+s}L^*. \tag{9}$$

In contrast, if $t(G_1) \leq \frac{t(G_{21})+t(G_{22})}{s}$, along with inequality (8), we obtain

$$\begin{aligned} L_1 = t(G_1) + t(G_{21}) &\leq \frac{(s+1)t(G_{21}) + t(G_{22})}{s} \\ &\leq \frac{1+2s}{s^2}t(G_{22}) = \frac{1+2s}{s}L_2. \end{aligned} \tag{10}$$

Thus, $L_1 + sL_2 \geq \frac{1+2s+s^2}{1+2s}L_1$. Following Lemma 2, we obtain $L^* \geq \frac{L_1+sL_2}{1+s}$. Therefore, we obtain

$$L^\sigma = L_1 \leq \frac{1+2s}{1+s}L^*. \tag{11}$$

Based on inequalities (9) and (11), we obtain that

$$\frac{L^\sigma}{L^*} \leq \max\left\{\frac{2+s}{1+s},\ \frac{1+2s}{1+s}\right\} \tag{12}$$

holds when $s \geq 1$.

Subcase 2.2.2: for $0 < s < 1$, Lemma 2 shows that

$$L_2 \leq \frac{(1+s)L^* - L_1}{s}. \tag{13}$$

Because $G_{21} \neq \varnothing$, the assignment of all jobs in G_{21} in the optimal schedule is considered. If all jobs in G_{21} are also assigned to machine M_1, we obtain $L^* \geq L_1 = L^\sigma$. In other words, schedule σ is also an optimal schedule and $L^\sigma = L^*$ in this case. Otherwise, there exists at least one job in G_{21} that is scheduled on machine M_2 in the optimal schedule. Let α be the set of all such jobs and job J_{k1} be the last-ranked maximum-size job in α (may be job J_{B1}) after sorting all jobs of α in nondecreasing order of size. We obtain $L^* \geq L_1 - t(\alpha)$, i.e.,

$$t(\alpha) \geq L_1 - L^*. \tag{14}$$

Lemma 3 shows that the completion time of job J_{k1} in σ is at least $t(\alpha) + p_{k1}$. Note that schedule σ is a Nash equilibrium and job J_{k1} cannot reduce its completion time by moving to a different machine. Thus, $L_2 + \frac{p_{k1}}{s} \geq t(\alpha) + p_{k1}$, along with inequalities (13) and (14), yields

$$\frac{(1+s)L^* - L_1}{s} + \frac{p_{k1}}{s} \geq L_1 - L^* + p_{k1}, \tag{15}$$

$$(1+2s)L^* + (1-s)p_{k1} \geq (1+s)L_1. \tag{16}$$

Because $0 < s < 1$ and $L^* \geq p_{k1}$, $(1+s)L_1 \leq (2+s)L^*$; thus, we obtain

$$\frac{L^\sigma}{L^*} \leq \max\left\{\frac{2+s}{1+s},\ \frac{1+2s}{1+s}\right\} \tag{17}$$

when $0 < s < 1$.

Based on inequalities (3), (6), (12), and (17), we obtain

$$L^\sigma \leq \min\left\{1+s,\ \max\left\{\frac{2+s}{1+s},\ \frac{1+2s}{1+s}\right\}\right\}L^* \tag{18}$$

if $L^\sigma = L_1 \geq L_2$.

Lemma 5 If $L^\sigma = L_2 > L_1$, then $L^\sigma \leq \min\{1+s,\ \max\{\frac{2+s}{1+s},\ \frac{1+2s}{1+s}\}\}L^*$.

Proof Because job $J_{B2} = (p_{B2},\ 2)$ is the largest job scheduled on machine M_2 in Nash equilibrium σ, we obtain

$$L_1 + p_{B2} \geq L_2. \tag{19}$$

If $s \geq 1$, Lemma 2 shows that $L_1 + sL_2 \leq (1+s)L^*$ and $p_{B2} \leq sL^*$. Along with inequality (19), we obtain

$$(1+s)L^\sigma = L_2 + sL_2 \leq sL_2 + L_1 + p_{B2}, \tag{20}$$

$$(1+s)L^\sigma \leq (1+2s)L^*. \tag{21}$$

If $0 < s < 1$, inequality (19) and Lemma 2 demonstrate that

$$L^\sigma = L_2 \leq L_1 + p_{B2} \leq L_1 + sL_2 \leq (1+s)L^*. \tag{22}$$

When $0 < s < 1$, inequality (19) demonstrates that

$$(1+s)L^\sigma = (1+s)L_2 \leq L_1 + sL_2 + p_{B2}. \tag{23}$$

Along with Lemma 2, we obtain

$$\frac{L^\sigma}{L^*} \leq \frac{2+s}{1+s}. \tag{24}$$

By combining inequalities (21), (22), and (24),

$$L^\sigma \leq \min\left\{1+s,\ \max\left\{\frac{2+s}{1+s},\ \frac{1+2s}{1+s}\right\}\right\}L^* \tag{25}$$

is ensured if $L^\sigma = L_2 > L_1$.

Lemmas 4 and 5 demonstrate that PoA for $Q2|\text{GoS}|C_{\max}$ is not greater than $f(s) = \min\{1+s,\ \max\{\frac{2+s}{1+s},\ \frac{1+2s}{1+s}\}\}$.

Theorem 1 Under the parallel processing policy, the upper PoA bound for the scheduling game $Q2|\text{GoS}|C_{\max}$ is given by

$$f(s) = \begin{cases} 1+s, & 0 < s \leq \frac{\sqrt{5}-1}{2}, \\ \frac{2+s}{1+s}, & \frac{\sqrt{5}-1}{2} < s \leq 1, \\ \frac{1+2s}{1+s}, & s > 1. \end{cases} \tag{26}$$

4 Lower bound of PoA

Theorem 2 Under the parallel processing policy, the lower PoA bound for the scheduling game $Q2|\text{GoS}|C_{\max}$ is given by Eq. (26).

Proof We construct instances to demonstrate that $f(s)$ is a lower bound of the PoA for the game (Table 1).

Instance for $s > 1$. Suppose that there is an instance comprising $\mathcal{J}$ with jobs $J_1 = \left(\frac{ks+k+s-1}{s},\ 1\right)$, $J_2 = J_3 = \cdots = J_{k+1} = (1,\ 2)$, and $J_{k+2} = J_{k+3} = \cdots = J_{k+1+\lfloor(k+1)s\rfloor} = \left(\frac{ks+s-1}{\lfloor(k+1)s\rfloor},\ 2\right)$, where k represents a positive integer and $\lfloor\cdot\rfloor$ denotes the round down. An optimal schedule assigns job J_1 to machine M_1 and all other $k + \lfloor(k+1)s\rfloor$ jobs to machine M_2. Then, we obtain $L^* = \frac{ks+k+s-1}{s}$. Next, we consider a schedule σ that assigns jobs $J_i,\ 1 \leq i \leq k+1$, to machine M_1 and the other $\lfloor(k+1)s\rfloor$ jobs $\left(\frac{ks+s-1}{\lfloor(k+1)s\rfloor},\ 2\right)$ to machine M_2. Note that the class of job J_1 is 1; it must select machine M_1. The completion time of job $J_i,\ 2 \leq i \leq k+1$, in σ is $k+1$ because $\frac{ks+k+s-1}{s} > 1$. If a job $J_i,\ 2 \leq i \leq k+1$, moves to a different machine, its completion time is $\frac{ks+s-1+1}{s} \geq k+1$. The completion time of job $J_i,\ k+2 \leq i \leq k+1+\lfloor(k+1)s\rfloor$, in σ is $\frac{ks+s-1}{s}$. If a job $J_i,\ k+2 \leq i \leq k+1+\lfloor(k+1)s\rfloor$, moves to a different machine, its completion time is $(k+2)\frac{ks+s-1}{\lfloor(k+1)s\rfloor}$. Note that $(k+2)\frac{ks+s-1}{\lfloor(k+1)s\rfloor} \geq \frac{ks+s-1}{s}$ because $\lfloor(k+1)s\rfloor \leq (k+2)s$. Therefore, each job cannot reduce its completion time by moving to a different machine. In other words, schedule σ is a Nash equilibrium that incurs the makespan $L^\sigma = \frac{2ks+k+s-1}{s}$. Therefore, we obtain

$$\frac{L^\sigma}{L^*} \geq \frac{2ks+k+s-1}{ks+k+s-1} \to \frac{1+2s}{1+s} \quad (k \to \infty). \tag{27}$$

Thus, the lower PoA bound for $Q2|\text{GoS}|C_{\max}$ is $\frac{1+2s}{1+s}$ when $s > 1$ under the parallel processing policy.

Instance for $\frac{\sqrt{5}-1}{2} < s \leq 1$. Suppose there is an instance comprising $\mathcal{J}$ with jobs $J_1 = (1+s,\ 2)$, $J_2 = (s^2+s-1,\ 2)$, and $J_3 = J_4 = J_5 = (1/3,\ 2)$. An optimal schedule assigns job J_1 to machine M_1 and the other four jobs to machine M_2. Then, we obtain $L^* = 1+s$. Next, we consider a schedule σ that assigns jobs J_1 and J_2 to machine M_2 and the other three jobs $(1/3,\ 2)$ to machine M_1. The completion time of job J_1 in σ is $2+s$. If job J_1 moves to machine M_1, its completion time remains $2+s$. The completion time of job J_2 in σ is $\frac{2(s^2+s-1)}{s}$. If job J_2 moves to machine M_1, its completion time is at least $\min\{4(s^2+s-1), s^2+s\}$, and $\frac{2(s^2+s-1)}{s} \leq \min\{4(s^2+s-1), s^2+s\}$ holds when $\frac{\sqrt{5}-1}{2} < s \leq 1$. The completion time of job $J_i,\ 3 \leq i \leq 5$, in σ is 1. If it moves to machine M_2, its completion time is at least $\min\left\{\frac{1}{s}, \frac{s^2+s-1+2/3}{s}\right\}$, and $1 \leq \min\left\{\frac{1}{s}, \frac{s^2+s-1+2/3}{s}\right\}$ holds when $\frac{\sqrt{5}-1}{2} < s \leq 1$. Therefore, each job cannot reduce its completion time by moving to a different machine. Therefore, schedule σ is a Nash equilibrium that incurs the makespan $L^\sigma = 2+s$. Thus, the lower PoA bound for $Q2|\text{GoS}|C_{\max}$ is $\frac{2+s}{1+s}$ when $\frac{\sqrt{5}-1}{2} < s \leq 1$.

Instance for $0 < s \leq \frac{\sqrt{5}-1}{2}$. Suppose that there is an instance comprising $\mathcal{J}$ with jobs $J_1 = (s,\ 2)$, $J_2 = (s+s^2,\ 2)$, and $J_3 = J_4 = \cdots = J_{\lceil\frac{1}{s}\rceil+2} = \left(\frac{1-s-s^2}{\lceil\frac{1}{s}\rceil},\ 1\right)$. Here, $\lceil\cdot\rceil$ denotes

Table 1 Tight instances for $Q2|\text{GoS}|C_{\max}$

Range of s	Nash equilibrium		Optimal solution		PoA
	Machine M_1	Machine M_2	Machine M_1	Machine M_2	
$0 < s \leq \frac{\sqrt{5}-1}{2}$	$(s,\ 2)$, $\lceil 1/s\rceil$ jobs $\left(\frac{1-s-s^2}{\lceil 1/s\rceil},\ 1\right)$	$(s+s^2,\ 2)^*$	$(s+s^2,\ 2)$, $\lceil 1/s\rceil$ jobs $\left(\frac{1-s-s^2}{\lceil 1/s\rceil},\ 1\right)$	$(s,\ 2)$	$1+s$
$\frac{\sqrt{5}-1}{2} < s \leq 1$	3 jobs $(1/3,\ 2)$	$(s^2+s-1,\ 2)^*$, $(1+s,\ 2)^*$	$(1+s,\ 2)$	$(s^s+s-1,\ 2)$, 3 jobs $(1/3,\ 2)$	$\frac{2+s}{1+s}$
$s > 1$	$\left(\frac{ks+k+s-1}{s},\ 1\right)^*$, k jobs $(1,\ 2)^*$	$\lfloor(k+1)s\rfloor$ jobs $\left(\frac{ks+s-1}{\lfloor(k+1)s\rfloor},\ 2\right)$	$\left(\frac{ks+k+s-1}{s},\ 1\right)$	k jobs $(1,\ 2)$, $\lfloor(k+1)s\rfloor$ jobs $\left(\frac{ks+s-1}{\lfloor(k+1)s\rfloor},\ 2\right)$	$\frac{1+2s}{1+s}$

* determines the makespan

the round up. An optimal schedule assigns job J_1 to machine M_2 and the other $1+\lceil\frac{1}{s}\rceil$ jobs to machine M_1. Then, we obtain $L^*=1$. Next, we consider a schedule σ that assigns job J_2 to machine M_2 and the other $1+\lceil\frac{1}{s}\rceil$ jobs to machine M_1. The completion time of job J_1 in σ is at most $1-s^2$. If job J_1 moves to machine M_2, its completion time is 2. Therefore, $1-s^2\leq 2$. The completion time of job J_2 in σ is $1+s$. If job J_2 moves to machine M_1, its completion time is also $1+s$ because $s+s^2>s>\frac{1-s-s^2}{\lceil\frac{1}{s}\rceil}$. Note that all other class 1 jobs must select machine M_1. Each job cannot reduce its completion time by moving to a different machine. Schedule σ is a Nash equilibrium that incurs the makespan $L^\sigma=1+s$. The lower PoA bound for $Q2|\text{GoS}|C_{\max}$ is $1+s$ when $0<s\leq\frac{\sqrt{5}-1}{2}$.

5 Conclusions

Following Theorems 1 and 2, we obtain the exact PoA for scheduling games on two uniform machines under the parallel processing policy.

Theorem 3 Under the parallel processing policy, the exact PoA for the scheduling game $Q2|\text{GoS}|C_{\max}$ is given by Eq. (26)

$s=1$ yields the exact PoA under the parallel processing policy for the selfish scheduling game on two identical parallel machines under a GoS provision.

Corollary 1 Under the parallel processing policy, the exact PoA for the scheduling game $P2|\text{GoS}|C_{\max}$ is 3/2.

In this study, we characterize the exact PoA for a two-machine setting with two hierarchical service levels. Extending our results to the general case of m uniform machines with two or more hierarchy levels constitutes an interesting open direction for future research. Our two-machine analysis reveals that the machine speed ratio s substantially complicates PoA derivation because case classification and tailored Nash equilibrium constructions are required based on different ranges of s. For the general m-machine setting, multiple hierarchical service levels and heterogeneous machine speeds introduce substantial obstacles to PoA characterization. Notably, even restricting attention to m uniform machines with only two hierarchy levels leaves a nontrivial open question. We consider a generalized m-machine hierarchical model structured as follows: There exist k $(1\leq k\leq m-1)$ machines, $M_1,M_2,\cdots,M_k$, with service certificate $g(M_t)=1$ $(1\leq t\leq k)$, which can schedule all jobs with identical unit speed $s_1=s_2=\cdots=s_k=1$. The remaining $m-k$ machines $M_{k+1},M_{k+2}\cdots,M_m$ possess service certificate $g(M_t)=2\,(k+1\leq t\leq m)$ and operate at uniform speed $s_{k+1}=\cdots=s_m=s$. Under this setup, we construct a concrete instance to establish a lower PoA bound, demonstrating that the PoA is at least $f(k,s)$, where

$$f(k,s)=\begin{cases}1+s, & 0<s\leq\frac{m-k-1}{m-k},\\ 2-\frac{1}{m-k}, & s>\frac{m-k-1}{m-k}.\end{cases}\tag{28}$$

Instance for m uniform machines is demonstrated as follows: Suppose that we construct an instance consisting of $\mathcal{J}$ with $(m-k)(m-k-1)$ jobs of type $(s,\ 2)$, one standalone special job $((m-k)s,\ 2)$, and k disjoint class 1 job subsets denoted as $\mathcal{J}_1,\mathcal{J}_2,\ldots,\mathcal{J}_k$. Let all $\mathcal{J}_i,1\leq i\leq k$, contain the same number of jobs with the same common size strictly smaller than s, while maintaining the total size $m-k$. An optimal schedule distributes all jobs from each $\mathcal{J}_i,1\leq i\leq k$, solely on machine $M_t,1\leq t\leq k$, assigns the special job $((m-k)s,\ 2)$ to machine M_{k+1}, and evenly allocates the $(m-k)(m-k-1)$ jobs $(s,\ 2)$ to $m-k-1$ machines $M_{k+2},M_{k+3},\cdots,M_m$. With this assignment, the optimal makespan satisfies $L^*=m-k$. Next, we analyze a feasible schedule σ constructed as follows: All jobs from subset $\mathcal{J}_i$ are assigned solely to machine M_t for each $1\leq t\leq k$; all $(m-k)(m-k-1)$ jobs $(s,2)$ are evenly split among $m-k$ machines $M_{k+1},M_{k+2},\ldots,M_m$; the standalone large job $((m-k)s,\ 2)$ is placed on M_{k+1} and M_1, if $s>\frac{m-k-1}{m-k}$ and $s\leq\frac{m-k-1}{m-k}$, respectively. We first verify that no class 1 job can benefit from migration: Each job set $\mathcal{J}_i,1\leq i\leq k$, is assigned to its dedicated machine with equal load $m-k$ across M_1 to M_k; thus, any cross-machine migration cannot decrease its completion time. Under schedule σ, the special job $((m-k)s,\ 2)$ is placed to minimize the equilibrium makespan $\min\{(m-k)+(m-k)s,(m-k-1)+(m-k)\}$. Relocating this special job to any alternative machine cannot decrease its individual completion time. If a job of type $(s,\ 2)$ moves to any machine among $\{M_1,M_2,\cdots,M_k\}$, its completion time is either $m-k+s$ or $m-k+2s$. The value $m-k+2s$ corresponds to the case that the job $(s,\ 2)$ moves to machine M_1 alongside $((m-k)s,\ 2))$ when $0<s\leq\frac{m-k-1}{m-k}$. Note that the completion time of a job $(s,\ 2)$ in σ is $m-k$ (when assigned alongside $((m-k)s,\ 2)$) or $m-k-1$; thus, a job of type $(s,\ 2)$ cannot benefit from migration because $m-k+2s>m-k+s>m-k>m-k-1$. Therefore, each job cannot reduce its completion time by moving to a different machine. The schedule σ is a Nash equilibrium that incurs the makespan $L^\sigma=\min\{(m-k)+(m-k)s,(m-k-1)+(m-k)\}$. Therefore, the PoA for this hierarchical game is at least $f(k,s)$.

The lower bound derived from the above constructed instance only partially resolves the generalized scheduling game. The tight exact PoA for arbitrary m uniform machines with GoS constraints remains an open research problem. Future work can complete the theoretical characterization of this model. Future research directions include deriving matching upper bounds, extending the proposed approach to multilevel job hierarchies, and evaluating the equilibrium efficiency of alternative scheduling mechanisms under various system optimization objectives, which can provide enriched theoretical insights into practical distributed resource allocation systems, such as cloud computing and network packet scheduling.

Acknowledgments

This work was supported by the Youth Foundation Project of the Humanities and Social Sciences Research Planning of the Ministry of Education (No. 20YJC630160).

Author contributions

Yong WU designed the research. Qi WEI analyzed the lower bound. Yong WU drafted the paper. Qi WEI revised and finalized the paper.

Conflict of interest

Both authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Declaration on the use of generative AI tools

During the preparation of this work, the authors used DeepSeek to improve language. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

Azar Y, Fleischer L, Jain K, et al., 2015. Optimal coordination mechanisms for unrelated machine scheduling. *Oper Res*, 63(3):489-500. https://doi.org/10.1287/opre.2015.1363

Bar-Noy A, Freund A, Naor J, 2001. On-line load balancing in a hierarchical server topology. *SIAM J Comput*, 31(2):527-549. https://doi.org/10.1137/s0097539798346135

Chen C, Xu YF, 2020. Coordination mechanisms for scheduling selfish jobs with favorite machines. *J Comb Optim*, 40(2):333-365. https://doi.org/10.1007/s10878-020-00592-2

Chen QQ, Tan ZY, 2021. Mixed coordination mechanisms for scheduling games on hierarchical machines. *Int Trans Oper Res*, 28(1):419-437. https://doi.org/10.1111/itor.12558

Christodoulou G, Koutsoupias E, Nanavati A, 2004. Coordination mechanisms. Proc 31st Int Colloquium on Automata, Languages and Programming, p.345-357. https://doi.org/10.1007/978-3-540-27836-8_31

Cohen J, Dürr C, Kim TN, 2011. Non-clairvoyant scheduling games. *Theor Comput Syst*, 49(1):3-23. https://doi.org/10.1007/s00224-011-9316-9

Cole R, Dodis Y, Roughgarden T, 2006. How much can taxes help selfish routing? *J Comput Syst Sci*, 72(3):444-467. https://doi.org/10.1016/j.jcss.2005.09.010

Czumaj A, Vöcking B, 2007. Tight bounds for worst-case equilibria. *ACM Trans Algorithms*, 3(1):4. https://doi.org/10.1145/1186810.1186814

Epstein L, 2010. Equilibria for two parallel links: the strong price of anarchy versus the price of anarchy. *Acta Inform*, 47(7-8):375-389. https://doi.org/10.1007/s00236-010-0124-5

Finn G, Horowitz E, 1979. A linear time approximation algorithm for multiprocessor scheduling. *BIT Numer Math*, 19(3):312-320. https://doi.org/10.1007/bf01930985

Guan L, Li JP, 2013. Coordination mechanism for selfish scheduling under a grade of service provision. *Inform Process Lett*, 113(8):251-254. https://doi.org/10.1016/j.ipl.2013.01.014

He CY, Tan ZY, 2026. Random coordination mechanism for scheduling games with machine modification. *Discrete Appl Math*, 382:108-123. https://doi.org/10.1016/j.dam.2025.11.038

Immorlica N, Li L, Mirrokni VS, et al., 2009. Coordination mechanisms for selfish scheduling. *Theor Comput Sci*, 410(17):1589-1598. https://doi.org/10.1016/j.tcs.2008.12.032

Koutsoupias E, Papadimitriou C, 1999. Worst-case equilibria. Proc 16th Symp on Theoretical Aspects of Computer Science, p.404-413. https://doi.org/10.1007/3-540-49116-3_38

Kress D, Meiswinkel S, Pesch E, 2018. Mechanism design for machine scheduling problems: classification and literature overview. *OR Spectrum*, 40(3):583-611. https://doi.org/10.1007/s00291-018-0512-8

Lin L, Tan ZY, 2014. Inefficiency of Nash equilibrium for scheduling games with constrained jobs: a parametric analysis. *Theor Comput Sci*, 521:123-134. https://doi.org/10.1016/j.tcs.2013.11.012

Lin L, Tan ZY, 2025. Price of anarchy of scheduling games on hierarchical machines with quadratic social cost. *Asia Pac J Oper Res*, 42(2):2450010. https://doi.org/10.1142/S0217595924500106

Muñoz FT, Linfati R, 2024. Bounding the price of anarchy of weighted shortest processing time policy on uniform parallel machines. *Mathematics*, 12(14):2223. https://doi.org/10.3390/math12142223

Roughgarden T, 2001. Stackelberg scheduling strategies. Proc 33rd Annual ACM Symp on Theory of Computing, p.104-113. https://doi.org/10.1145/380752.380783

Wan L, Deng XF, Tan ZY, 2013. Inefficiency of Nash equilibria with parallel processing policy. *Inform Process Lett*, 113(13):465-469. https://doi.org/10.1016/j.ipl.2013.03.006

Wu Y, Cheng TCE, Ji M, 2015. Inefficiency of the Nash equilibrium for selfish machine covering on two hierarchical uniform machines. *Inform Process Lett*, 115(11):838-844. https://doi.org/10.1016/j.ipl.2015.06.005

Yu LF, She K, Gong HG, et al., 2010. Price of anarchy in parallel processing. *Inform Process Lett*, 110(8-9):288-293. https://doi.org/10.1016/j.ipl.2010.02.003