Scheduling Games with Machine-Dependent Priority Lists

Marc Schröder[1], Tami Tamir[2]*, and Vipin Ravindran Vijayalakshmi[1]**

[1] Chair of Management Science, RWTH Aachen, Germany
{marc.schroeder,vipin.rv}@oms.rwth-aachen.de
[2] School of Computer Science, The Interdisciplinary Center, Israel
tami@idc.ac.il

Abstract. We consider a scheduling game in which jobs try to minimize their completion time by choosing a machine to be processed on. Each machine uses an individual priority list to decide on the order according to which the jobs on the machine are processed. We characterize four classes of instances in which a pure Nash equilibrium (NE) is guaranteed to exist, and show by means of an example, that none of these characterizations can be relaxed. We then bound the performance of Nash equilibria for each of these classes with respect to the makespan of the schedule and the sum of completion times. We also analyze the computational complexity of several problems arising in this model. For instance, we prove that it is NP-hard to decide whether a NE exists, and that even for instances with identical machines, for which a NE is guaranteed to exist, it is NP-hard to approximate the best NE within a factor of $2 - \frac{1}{m} - \epsilon$ for every $\epsilon > 0$.
In addition, we study a generalized model in which players' strategies are subsets of resources, where each resource has its own priority list over the players. We show that in this general model, even unweighted symmetric games may not have a pure NE, and we bound the price of anarchy with respect to the total players' costs.

Keywords: Scheduling Games · Priority Lists · Price of Anarchy.

1 Introduction

Scheduling problems have traditionally been studied from a centralized point of view in which the goal is to find an assignment of jobs to machines so as to minimize some global objective function. Two of the classical results are that Smith's rule, i.e., schedule jobs in decreasing order according to their ratio of weight over processing time, is optimal for single machine scheduling with the sum of weighted completion times as the objective [23], and list scheduling, i.e., greedily assign the job with the highest priority to a free machine,

* This research is supported by The Israel Science Foundation (ISF). Grant No. 1036/17.
** This work is supported by the German research council (DFG) Research Training Group 2236 UnRAVeL

yields a 2-approximation for identical machines with the minimum makespan objective [14]. Many modern systems provide service to multiple strategic users, whose individual payoff is affected by the decisions made by others. As a result, non-cooperative game theory has become an essential tool in the analysis of job-scheduling applications. The jobs are controlled by selfish users who independently choose which resources to use. Job-scheduling games have by now been widely studied and many results regarding the inefficiency of equilibria in different settings are known.

A particular focus has been placed on finding coordination mechanisms [6], i.e., local scheduling policies, that induce a good system performance. In fact, recently Caragiannis et al. [4] proposed a framework that uses such policies to come up with combinatorial approximation algorithms for the underlying optimization problem. It is common to assume that ties are broken in a consistent manner (see, e.g., Immorlica et al. [17]), or that there are no ties at all (see, e.g., Cole et al. [7]). In practice, there is no real justification for this assumption, except that it avoids subtle difficulties in the analysis. In this paper we relax this restrictive assumption and consider the more general setting in which machines have arbitrary individual priority lists. That is, each machine schedules those jobs that have chosen it according to its priority list. The priority lists are publicly known to the jobs.

In this paper we analyze the effect of having machine-dependent priority lists on the corresponding job-scheduling game. We study the existence of Nash equilibrium, the complexity of identifying whether a NE profile exists, the complexity of calculating a NE, in particular a good one, and the equilibrium inefficiency.

1.1 The Model

An instance of a *scheduling game with machine-dependent priority lists* is given by a tuple $G = \langle N, M, (w_i)_{i\in N}, (c_j)_{j\in M}, (\pi_j)_{j\in M}\rangle$, where N is a finite set of $n \geq 1$ jobs, M is a finite set of $m \geq 1$ machines, $w_i \in \mathbb{R}_+$ is the weight of job $i \in N$, $c_j \in \mathbb{R}_+$ is the processing delay of machine $j \in M$, and $\pi_j : N \to \{1, \dots, n\}$ is the priority list of machine $j \in M$. In the literature, it is common to characterize the jobs by their *processing time* and the machines by their *speed*. We prefer to refer to weight instead of to processing time, and to delay, which is the inverse of speed, in order to be consistent with the general definition of congestion games.

A strategy profile $s = (s_i)_{i\in N}$ assigns a machine $s_i \in M$ to every job $i \in N$. Given a strategy profile s, the jobs are processed according to their order in the machines' priority lists. The set of jobs that delay job i in s is $B_i(s) = \{i' \in N | s_{i'} = s_i \wedge \pi_{s_i}(i') \leq \pi_{s_i}(i)\}$. Note that job i itself also belongs to $B_i(s)$. Let $w^i(s) = \sum_{i'\in B_i(s)} w_{i'}$. The cost of job $i \in N$ is equal to its completion time in s, given by $cost_i(s) = c_j \cdot w^i(s)$.

A more general model is that of a *congestion game with resource-dependent priority lists*, in which the strategy space of a player consists of *subsets of resources*. Formally, an instance of the general game is given by a tuple $G = \langle N, E, (\Sigma_i)_{i\in N}, (w_i)_{i\in N}, (c_e)_{e\in E}, (\pi_e)_{e\in E}\rangle$, where N is a finite set players, E is a finite set of resources, $\Sigma_i \subseteq 2^E$ is the set of feasible strategies for player $i \in N$, $w_i \in \mathbb{R}_+$ is the weight of player $i \in N$, $c_e \in \mathbb{R}_+$ is the cost coefficient of resource $e \in E$, and $\pi_e : N \to \{1, \dots, n\}$ is the priority list of resource E that defines its preference over the players using it.

Scheduling games are symmetric singleton congestion games in which the strategy space of each job is the set of all machines. For the general setting, the players' costs are defined as follows. Given a strategy profile $s = (s_i)_{i \in N} \in \times_{i \in N} \Sigma_i$, for every player $i \in N$, and resource $e \in s_i$, let $B_{ie}(s) = \{i' \in N \mid e \in s_{i'} \wedge \pi_e(i') \leq \pi_e(i)\}$, and define $w_e^i(s) = \sum_{i' \in B_{ie}(s)} w_{i'}$. The cost of a player $i \in N$ is given by, $cost_i(s) = w_i \cdot \sum_{e \in s_i} c_e \cdot w_e^i(s)$.

Notice that for general congestion games, we assume that players' costs are multiplied by their weight, whereas we do not make that assumption for scheduling games. This has no implications for the existence of Nash equilibria, but only affects the efficiency result.

Each job chooses a strategy so as to minimize its cost. A strategy profile s is a *pure Nash equilibrium (NE)* if for all $i \in N$ and all $s'_i \in \Sigma_i$, we have $cost_i(s) \leq cost_i(s'_i, s_{-i})$. Let $\mathcal{E}(G)$ denote the set of Nash equilibria for a given instance G. Notice that $\mathcal{E}(G)$ may be empty.

For a profile s, let $cost(s)$ denote the cost of s. The cost is defined with respect to some objective. For example, the total players' cost or the maximum cost of a player. It is well known that decentralized decision-making may lead to sub-optimal solutions from the point of view of the society as a whole. For a game G, let $P(G)$ be the set of feasible profiles of G. We denote by $OPT(G)$ the cost of a social optimal (SO) solution; i.e., $OPT(G) = \min_{s \in P(G)} cost(s)$. We quantify the inefficiency incurred due to self-interested behavior according to the *price of anarchy* (PoA) [19] and *price of stability* (PoS) [2] measures. The PoA is the worst-case inefficiency of a pure Nash equilibrium, while the PoS measures the best-case inefficiency of a pure Nash equilibrium.

Definition 1. *Let $\mathcal{G}$ be a family of games, and let G be a game in $\mathcal{G}$. Let $\mathcal{E}(G)$ be the set of pure Nash equilibria of the game G. Assume that $\mathcal{E}(G) \neq \emptyset$.*

- *The* price of anarchy *of G is the ratio between the* maximal *cost of a NE and the social optimum of G. That is, $PoA(G) = \max_{s \in \mathcal{E}(G)} cost(s)/OPT(G)$. The* price of anarchy *of the family of games $\mathcal{G}$ is $PoA(\mathcal{G}) = sup_{G \in \mathcal{G}} PoA(G)$.*
- *The* price of stability *of G is the ratio between the* minimal *cost of a NE and the social optimum of G. That is, $PoS(G) = \min_{s \in \mathcal{E}(G)} cost(s)/OPT(G)$. The* price of stability *of the family of games $\mathcal{G}$ is $PoS(\mathcal{G}) = sup_{G \in \mathcal{G}} PoS(G)$.*

1.2 Our Contribution

We start by studying scheduling games, i.e., each job has to choose one machine to be processed on, and then based on the choices of the jobs, each machine schedules the jobs according to its individual priority list. We first show that a Nash equilibrium in general need not exist, and use this example to show that it is NP-complete to decide whether a particular game has a Nash equilibrium. We then extend known results in order to provide a characterization of instances in which a pure Nash equilibrium is guaranteed to exist. Specifically, existence is guaranteed if the game belongs to at least one of the following four classes: $\mathcal{G}_1$: all jobs have the same weight, $\mathcal{G}_2$: there are two machines, $\mathcal{G}_3$: all machines have the same processing delay (shown in [9]), and $\mathcal{G}_4$: all machines have the same priority list (shown in [11]). For all four of these classes, there is a polynomial time algorithm that computes a Nash equilibrium. In fact, if jobs are unweighted, better-response dynamics converge to a Nash equilibrium in polynomial time. This characterization is tight in a sense that our inexistence example disobeys it in a minimal way: it describes a game on three machines, two of them having the same processing delay and the same priority list.

We analyze the inefficiency of Nash equilibria by means of two different measures of efficiency: the makespan, i.e., the maximum completion time of a job, and the sum of completion times. For all four classes of games with a guaranteed Nash equilibrium we provide tight bounds for the price of anarchy and the price of stability with respect to both measures. Our results are summarized in Table 1. For two machines with processing delays $c_1 = 1$ and $c_2 = c \geq 1$, we prove that the PoA and the PoS are at most $1 + \frac{c}{c+1}$ if $c \leq \frac{\sqrt{5}+1}{2}$, and $1 + \frac{1}{c}$ if $c \geq \frac{\sqrt{5}+1}{2}$. Our analysis is tight for all $c \geq 1$. The maximal inefficiency, listed in Table 1, is achieved with $c = \frac{\sqrt{5}+1}{2}$.

Instance class \ Objective	Makespan		Sum of Completion Times	
	PoA	PoS	PoA	PoS
$\mathcal{G}_1$: Unweighted jobs	1	1	1	1
$\mathcal{G}_2$: Two machines	$(\sqrt{5}+1)/2$	$(\sqrt{5}+1)/2$	$\Theta(n)$	$\Theta(n)$
$\mathcal{G}_3$: Identical machines	$2 - 1/m$	$2 - 1/m$	$\Theta(n/m)$	$\Theta(n/m)$
$\mathcal{G}_4$: Global priority list	$\Theta(m)$	$\Theta(m)$	$\Theta(n)$	$\Theta(n)$

Table 1. Our results for the equilibrium inefficiency.

In terms of computational complexity, we prove that it is NP-hard to approximate the best NE within a factor of $2 - \frac{1}{m} - \epsilon$ for all $\epsilon > 0$, if machines have identical processing delays and the minimum makespan objective is considered. Recall that $2 - \frac{1}{m}$ is the price of anarchy for these instances. In particular, this implies that the simple greedy algorithm that computes a Nash equilibrium by letting each machine with the current least load get the most preferred unassigned job, is the best one can hope for.

We finally generalize the model to allow for arbitrary strategy sets. We show that in general, even with unweighted jobs, a Nash equilibrium need not exist by making use of the famous Condorcet paradox [8]. We then use this example to prove that the question whether a Nash equilibrium exists is NP-hard, even with unweighted jobs. We lastly study the price of anarchy with respect to the sum of weighted costs and show that the upper bound of 4 proven by Cole et al. [7]

for unrelated machine scheduling with Smith's rule also extends to congestion games with resource-dependent priority lists. This ratio is smaller than the price of anarchy of the atomic game with priorities defined by Farzad et al. [11].

Due to space constraints, some of the proofs are omitted. A full version of this paper with all the proofs can be found at https://arxiv.org/abs/1909.10199.

1.3 Related Work

Scheduling games. The existence and inefficiency of Nash equilibria in scheduling games gained lots of attention over the recent years. We refer to Vöcking [24] for a recent overview. For existence, Immorlica et al. [17] proved that for unrelated machines, i.e., different machines can have different processing times for jobs, and priority lists based on shortest processing time first with consistent tie-breaking, the set of Nash equilibria is always non-empty and corresponds to the set of solutions of the Ibarra-Kim algorithm [16].

The standard measure for the inefficiency of Nash equilibria is the price of anarchy [19]. This measure has been widely studied for different measures of efficiency. Most attention has been addressed on minimizing the makespan. Czumaj and Vöcking [10] gave tight bounds for related machines that grow as the number of machines grows, whereas Awerbuch et al. [3] and Gairing et al. [12] provided tight bounds for restricted machine settings. An alternative measure of efficiency is utilitarian social welfare, that is, the sum of weighted completion times. Correa and Queyranne [9] proved a tight upper bound of 4 for restricted related machines with priority lists derived from Smith's rule. Cole et al. [7] generalized the bound of 4 to unrelated machines with Smith's rule. Hoeksma and Uetz [15] gave a tighter bound for the more restricted setting in which jobs have unit weights and machines are related.

Congestion games with priorities. Rosenthal [21] proved that congestion games are potential games and thus have a pure Nash equilibrium. Ackermann et al. [1] were the first to study a congestion game with priorities. They proposed a model in which users with higher priority on a resource displace users with lower priority. Similar to our model, Farzad et al. [11] studied priority based selfish routing for non-atomic and atomic users. Gourvès et al. [13] studied capacitated congestion games to characterize the existence of pure Nash equilibria and computation of an equilibrium when they exist. Piliouras et al. [20] assumed that the priority lists are unknown to the players a priori and consider different risk attitudes towards having a uniform at random ordering.

2 Equilibrium Existence and Computation

In this section we give a precise characterization of instances that are guaranteed to have a NE. The conditions that we provide are sufficient but not necessary. A natural question is to decide whether a given game instance that does not fulfill any of the conditions has a NE. We show that answering this question is a NP-complete problem.

We first show that a NE may not exist, even with only three machines, two of which have the same delay and the same priority list.

Example 1. Consider the game G^* with 5 jobs, $N = \{a, b, c, d, e\}$, and three machines, $M = \{M_1, M_2, M_3\}$, with $\pi_1 = (a, b, c, d, e)$, and $\pi_2 = \pi_3 = (e, d, b, c, a)$. The first machine has delay $c_1 = 1$ while the two other machines have delay $c_2 = c_3 = 2$. The job weights are $w_a = 5, w_b = 4, w_c = 4 + 2\epsilon, w_d = 9 + \epsilon$, and $w_e = 2$, where $\epsilon > 0$ but small.

Job a is clearly on M_1 in every NE. It is easy to see that in every NE at least one of b, c and d is on M_1. Therefore, job e is first on M_2 or M_3. Since these two machines have the same priority list and the same delay function, we can assume w.l.o.g., that if a NE exists, then there exists a NE in which job e is on M_3. We show that no NE exists by considering the three possible strategies of job b.

1. b is on M_1: If d is not on M_2 or M_3, then b prefers M_2 to M_1. If d is on M_2, then c is on M_3 (since $12 + 4\epsilon < 13 + 2\epsilon$). As a result, d prefers M_1 (since $18 + \epsilon < 18 + 2\epsilon$), so b prefers M_2. Finally, given that e is on M_3, d is not on M_3.
2. b is on M_2: job c prefers M_1, where it completes at time $9+2\epsilon$, while after e on M_3 it completes at time $12+4\epsilon$. Now d prefers M_2, (since $18+2\epsilon < 18+3\epsilon$). So b prefers M_1.
3. b is on M_3: Being after e, job b prefers M_1.

Thus, the game G^* has no pure Nash equilibrium.

We can use the above example to show that deciding whether a game instance has a NE is NP-complete by using a reduction from 3-bounded 3-dimensional matching. The proof is omitted. A more involved hardness proof that uses a similar technique is given in the proof of Theorem 12.

Theorem 1. *Given an instance of a scheduling game, it is NP-complete to decide whether the game has a NE.*

Our next results are positive. When combined with known results regarding equilibrium existence, and our example above, we get a tight characterization of classes of instances with a guaranteed Nash equilibrium.

The following algorithm is intended for instances in the class $\mathcal{G}_1$, that is, for all $i \in N$, $w_i = 1$. It assigns the jobs greedily, where in each step, a job is added on a machine on which the cost of the next job is minimal.

Algorithm 1 Calculating a NE of unit-weight jobs on related machines

```
Let ℓ_j denote the number of jobs assigned on machine j. Initially, ℓ_j = 0 for all 1 ≤ j ≤ m.
repeat
    Let j* = arg min_j  c_j · (ℓ_j + 1).
    Assign on machine j* the first unassigned job on its priority list.
    ℓ_{j*} = ℓ_{j*} + 1.
until all jobs are scheduled
```

Theorem 2. *If $w_i = 1$ for all jobs $i \in N$, then Algorithm 1 calculates a NE.*

In fact, for the unweighted case, every sequence of better responses converges in polynomial time. Given a strategy profile s, a strategy s'_i for job $i \in N$ is a better response if $cost_i(s'_i, s_{-i}) < cost_i(s)$. The proof of the following theorem is omitted, but analyzes a potential function that is introduced by Gourvès et al. [13].

Theorem 3. *If $w_i = 1$ for all jobs $i \in N$, then jobs reach an equilibrium after polynomially many better response moves.*

Our next result considers the number of machines and completes the picture. Since our inexistence example uses three machines, out of which two are identical (in both delay and priority list), we cannot hope for a wider positive result.

Theorem 4. *If $m \leq 2$, then a NE exists and can be calculated efficiently.*

Proof. For a single machine, the priority list defines the only feasible schedule, which is clearly a NE. For $m = 2$, assume w.l.o.g., that $c_1 = 1$ and $c_2 = c \geq 1$. Consider the following algorithm, which initially assigns all the jobs on the fast machine. Then, the jobs are considered according to their order in π_2, and every job gets an opportunity to migrate to M_2.

Algorithm 2 Calculating a NE schedule on two related machines

1: Assign all the jobs on M_1 (the fast machine) according to their order in π_1.
2: For $1 \leq k \leq n$, let the job i for which $\pi_2(i) = k$ perform a best-response move (migrate to M_2 if this reduces its completion time).

Denote by s^1 the schedule after the first step of the algorithm (where all the jobs are on M_1), and let s denote the schedule after the algorithm terminates. We show that s is a NE.

Claim. No job for which $s_i = 1$ has a beneficial migration.

Proof. Assume by contradiction that job i is assigned on M_1 and has a beneficial migration. Assume that $\pi_2(i) = k$. Job i was offered to perform a migration in the k-th iteration of step 2 of the algorithm, but chose to remain on M_1. The only migrations that took place after the k-th iteration are from M_1 to M_2. Thus, if migrating is beneficial for i after the algorithm completes, it should have been beneficial also during the algorithm, contradicting its choice to remain on M_1.

Claim. No job for which $s_i = 2$ has a beneficial migration.

Proof. Assume by contradiction that the claim is false and let i be the first job on M_2 (first with respect to π_2) that may benefit from returning to M_1. Let s^1 denote the schedule before job i migrates to M_2 - during the second step of the algorithm. Recall that $cost_i(s)$ is the completion time of job i on M_2, and $cost_i(s^1)$ is its completion time on M_1 before its migration.

Since the jobs are activated according to π_2 in the 2-nd step of the algorithm, no jobs are added before job i on M_2. Job i may be interested in returning to M_1 only if some jobs that were processed before it on M_1, move to M_2 after its migration. Denote by Δ the set of these jobs, and let δ be their total weight. Let i' be the last job from Δ to complete its processing in s. Since job i' performs its migration out of M_1 after job i, and jobs do not join M_1 during step 2 of the algorithm, the completion time of i' when it performs the migration is at most $cost_{i'}(s^1)$. The migration from M_1 to M_2 is beneficial for i', thus, $cost_{i'}(s) < cost_{i'}(s^1)$.

The jobs in Δ are all before job i in π_1 and after job i in π_2. Therefore, $cost_{i'}(s^1) < cost_i(s^1)$, and $cost_{i'}(s) \geq cost_i(s) + c\delta$. Finally, we assume that s is not stable and i would like to return to M_1. By returning, its completion time would be $cost_i(s^1) - \delta$. Given that the migration is beneficial for i, and that i is the first job who likes to return to M_2, we have that $cost_i(s^1) - \delta < cost_i(s)$.

Combining the above inequalities, we get

$$cost_i(s^1) < cost_i(s)+\delta \leq cost_{i'}(s)-(c-1)\delta < cost_{i'}(s^1)-(c-1)\delta < cost_i(s^1)-(c-1)\delta.$$

This contradicts the fact that $c \geq 1$ and $\delta \geq 0$.

By combining the two claims, we conclude that s is a NE. □

3 Equilibrium Inefficiency

Two common measures for evaluating the quality of a schedule are the makespan, given by $C_{max}(s) = \max_{i \in N} cost_i(s)$, and the sum of completion times, given by $\sum_{i \in N} cost_i(s)$. In this section we analyze the equilibrium inefficiency with respect to each of the two objectives, for each of the four classes for which a NE is guaranteed to exist.

We begin with $\mathcal{G}_1$, the class of instances with unweighted jobs. For this class we show that allowing arbitrary priority lists does not hurt the social cost, even on machines with different speeds.

Theorem 5. $PoA(\mathcal{G}_1) = PoS(\mathcal{G}_1) = 1$ *for both the min-makespan and the sum of completion times objective.*

In Theorem 4 it is shown that a NE exists for any instance on two related machines. We now analyze the equilibrium inefficiency of this class. Let $\mathcal{G}_2^c$ denote the class of games played on two machines with delays $c_1 = 1$ and $c_2 = c \geq 1$.

Theorem 6. *For the min-makespan objective,* $PoA(\mathcal{G}_2^c) = PoS(\mathcal{G}_2^c) = 1 + \frac{1}{c}$ *if* $c \geq \frac{\sqrt{5}+1}{2}$*, and* $PoA(\mathcal{G}_2^c) = PoS(\mathcal{G}_2^c) = 1 + \frac{c}{c+1}$ *if* $c \leq \frac{\sqrt{5}+1}{2}$*.*

Proof. Let $G \in \mathcal{G}_2^c$. Let $W = \sum_i w_i$ be the total weight of all jobs. Assume first that $c \geq \frac{\sqrt{5}+1}{2}$. For the minimum makespan objective, $OPT(G) \geq W/(1+1/c)$. Also, for any NE s, we have that $C_{max}(s) \leq W$, since every job can migrate to be last on the fast machine and have completion time at most W. Thus, PoA $\leq 1 + 1/c$.

Assume next that $c < \frac{\sqrt{5}+1}{2}$. Let job a be the last job to complete in a worst Nash equilibrium s, w_1 be the total weight of all jobs different from a on machine 1, and w_2 be the total weight of all jobs different from a on machine 2 in s. Then since s is a Nash equilibrium, $C_{max}(s) \leq w_1 + w_a$ and $C_{max}(s) \leq c \cdot (w_2 + w_a)$. Combining these two inequalities yields

$$C_{max}(s) \leq \frac{W + w_a}{1 + \frac{1}{c}} \leq (1 + c/(c+1)) \cdot OPT(G),$$

where for the inequality we use that $OPT(G) \geq W/(1+1/c)$ and $OPT(G) \geq w_a$, and thus PoA$\leq 1 + c/(c+1)$.

For the PoS lower bound, assume first that $c > \frac{\sqrt{5}+1}{2}$. Consider an instance consisting of two jobs, a and b, where $w_a = 1$ and $w_b = c$. The priority lists are $\pi_1 = \pi_2 = (a, b)$. The unique NE is that both jobs are on the fast machine. $cost_a(s) = 1, cost_b(s) = c + 1$. For every $c > \frac{\sqrt{5}+1}{2}$, it holds that $c + 1 < c^2$, therefore, job b does not have a beneficial migration. An optimal schedule assigns job a on the slow machine, and both jobs complete at time c. The corresponding PoS is $\frac{c+1}{c} = 1 + \frac{1}{c}$. [3]

Assume now that $c < \frac{\sqrt{5}+1}{2}$. Consider an instance consisting of three jobs, x, y and z, where $w_x = 1, w_y = \frac{1+c-c^2}{c^2}$, and $w_z = \frac{1+c}{c}$. The priority lists are $\pi_1 = \pi_2 = (x, y, z)$. Note that $w_y \geq 0$ for every $c \leq \frac{\sqrt{5}+1}{2}$. The unique NE is when jobs x and z are on the fast machine, and job y on the slow machine. Indeed, job y prefers being alone on the slow machine since $\frac{1+c}{c^2} > \frac{1+c-c^2}{c}$. Job z prefers joining x on the fast machine since $1 + w_z < c(w_y + w_z)$. The makespan is $1 + w_z = \frac{1+2c}{c}$. In an optimal schedule, job z is alone on the fast machine, and jobs x and y are on the slow machine. Both machines have the same completion time $\frac{1+c}{c}$. The PoS is $\frac{1+2c}{1+c} = 1 + \frac{c}{c+1}$. □

Theorem 7. *For the sum of completion times objective,* $PoA(\mathcal{G}_2^c) = \Theta(n)$ *and* $PoS(\mathcal{G}_2^c) = \Theta(n)$ *for all* $c \geq 1$.

We turn to analyze the equilibrium inefficiency of the class $\mathcal{G}_3$, consisting of games played on identical-speed machines, having machine-based priority lists. The proof of the following theorem is based on the observation that every NE schedule is a possible outcome of Graham's *List-scheduling* (LS) algorithm [14].

Theorem 8. *For the min-makespan objective,* $PoA(\mathcal{G}_3)$ =*PoS* $(\mathcal{G}_3) = 2 - \frac{1}{m}$.

Theorem 9. *For the sum of completion times objective,* $PoA(\mathcal{G}_3) \leq \frac{n-1}{m} + 1$, *and for every* $\epsilon > 0$, $PoS(\mathcal{G}_3) \geq \frac{n}{m} - \epsilon$.

[3] For $c = \frac{\sqrt{5}+1}{2}$, by taking $w_b = c + \epsilon$, the PoS approaches $1 + c/(c+1)$ as $\epsilon \to 0$.

The last class of instances for which a NE is guaranteed to exist includes games with a global priority list, and is denoted by $\mathcal{G}_4$. It is easy to verify that for this class, the only NE profiles are those produced by List-Scheduling algorithm, where the jobs are considered according to their order in the priority list. Different NE may be produced by different tie-breaking rules. Thus, the equilibrium inefficiency is identical to the approximation ratio of LS [5]. Since the analysis of LS is tight, this is also the PoS.

Theorem 10. *For the min-makespan objective, $PoS(\mathcal{G}_4) = PoA(\mathcal{G}_4) = \Theta(m)$.*

For the sum of completion times objective, we note that the proof of Theorem 7 for two related machines uses a global priority list. The analysis of the PoA is independent of the number and delays of machines.

Theorem 11. *For the sum of completion times objective, $PoA(\mathcal{G}_4) = \Theta(n)$ and $PoS(\mathcal{G}_4) = \Theta(n)$.*

3.1 Hardness of Approximating the Minimum Makespan NE

Correa and Queyranne [9] showed that if all the machines have the same speeds, but arbitrary priority lists, then a NE is guaranteed to exist, and can be calculated by a simple greedy algorithm. In Theorem 8, we have shown that the PoA is at most $2 - \frac{1}{m}$. In this subsection, we show that we cannot hope for a better algorithm than the simple greedy algorithm. More formally, we prove that it is NP-hard to approximate the best NE within a factor of $2 - \frac{1}{m} - \epsilon$ for all $\epsilon > 0$.

Theorem 12. *If for all machines $c_j = 1$, then it is NP-hard to approximate the best NE w.r.t. the makespan objective within a factor of $2 - \frac{1}{m} - \epsilon$ for all $\epsilon > 0$.*

Proof. We show that for every $\epsilon > 0$, there is an instance on m identical machines for which it is NP-hard to distinguish whether the game has a NE profile with makespan at most $m + 2\epsilon$ or at least $2m - 1$.

The hardness proof is by a reduction from 3-bounded 3-dimensional matching (3DM-3). The input to the 3DM-3 problem is a set of triplets $T \subseteq X \times Y \times Z$, where $|X| = |Y| = |Z| = n$. The number of occurrences of every element of $X \cup Y \cup Z$ in T is at most 3. The number of triplets is $|T| \geq n$. The goal is to decide whether T has a 3D-matching of size n, i.e., there exists a subset $T' \subseteq T$, such that $|T'| = n$, and every element in $X \cup Y \cup Z$ appears exactly once in T'. 3DM-3 is known to be NP-hard [18].

Given an instance of 3DM-3 and $\epsilon > 0$, consider the following game on $m = |T| + 2$ machines, $M_1, M_2, \ldots, M_{|T|+2}$. The set of jobs includes job a of weight m, job b of weight $m - 1$, a set D of $|T| - n$ dummy jobs of weight 3ϵ, two dummy jobs d_1, d_2 of weight 2ϵ, a set U of $(m-1)^2$ unit-weight jobs, and $3n$ jobs of weight ϵ - one for each element in $X \cup Y \cup Z$.

We turn to describe the priority lists. When the list includes a set, it means that the elements can appear in an arbitrary order. For the first machine, $\pi_1 = (d_1, b, a, U, X, Y, Z, D, d_2)$. For the second machine, $\pi_2 = (d_2, X, Y, Z, b, U, a, d_1)$. The $m - 2$ right machines are *triplet-machines*. For every $t = (x_i, y_j, z_k) \in T$, the priority list of the triplet-machine corresponding to t is $(D, x_i, y_j, z_k, U, X \setminus \{x_j\}, Y \setminus \{y_j\}, Z \setminus \{z_j\}, d_1, d_2, a, b)$.

The heart of the reduction lies in determining the priority lists. The idea is that if a 3D-matching exists, then job b would prefer M_2 and let job a be assigned early on M_1. However, if there is no 3D-matching, then some job originated from the elements in $X \cup Y \cup Z$ will precede job b on M_2, and b's best-response would be on M_1. The jobs in U have higher priority than job a on all the machines except for M_1, thus, unless job a is on M_1, it is assigned after $|U|/(m-1)$ unit-jobs from U, inducing a schedule with high makespan.

Observe that in any NE, the two dummy jobs of weight 2ϵ are assigned as the first jobs on M_1 and M_2. Also, the dummy jobs in D have the highest priority on the triplet-machines, thus, in every NE, there are $|D| = |T| - n$ triplet-machines on which the first job is from D.

The following two claims complete the proof. Figure 1 provides an example for $m = 5$.

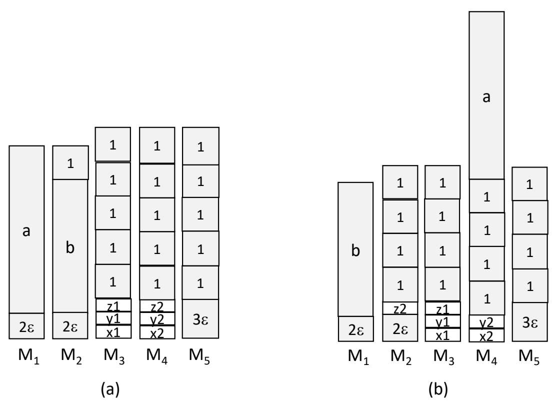

Fig. 1. (a) A NE schedule for $n = 2$ and $T = \{(x_1, y_1, z_1), (x_2, y_2, z_2), (x_1, y_2, z_2)\}$. A matching of size 2 exists. The makespan is $5 + 3\epsilon$. (b) A NE schedule for $T = \{(x_1, y_1, z_1), (x_2, y_2, z_1), (x_1, y_2, z_2)\}$. A matching of size 2 does not exist. The makespan is $9 + 2\epsilon$.

Claim. If a 3D-matching of size n exists, then the game has a NE schedule whose makespan is $m + 2\epsilon$.

Proof. Let T' be a matching of size n. Assign the jobs of $X \cup Y \cup Z$ on the triplet-machines corresponding to T' and the jobs of D on the remaining triplet-machines. Also, assign d_1 and d_2 on M_1 and M_2 respectively. M_1 and M_2 now have load 2ϵ while the triplet machines have load 3ϵ. Next, assign job a on M_1 and job b on M_2. Finally, add the unit-jobs as balanced as possible: m jobs on each triplet-machine and a single job after job b on M_2. It is easy to verify that the resulting assignment is a NE. Its makespan is $m + 3\epsilon$.

Claim. If a 3D-matching of size n does not exist then every NE schedule has makespan at least $2m - 1$.

Proof. Let s be a NE profile of an instance for which a matching does not exist. From the above observations, there are exactly n triplet-machines on which the first element is not from D. Since a matching does not exist, for at least one such machine, there are at most two jobs from $X \cup Y \cup Z$ whose priority is higher than the priority of the unit jobs. Thus, at least one job from $X \cup Y \cup Z$ prefers M_2, and is assigned after d_2. As a result, job b prefers M_1, where it can start being processed at time 2ϵ. Given that job b is on M_1, and that there are at least $m - 1$ unit jobs on each machine, job a cannot start its processing earlier than $m - 1$, implying that its completion time is at least $2m - 1$.

□

4 General Congestion Games with Priority Lists

In this last section we consider a generalization of the model that allows for arbitrary strategy sets. First, we show that a Nash equilibrium need not exist and in fact, the question whether a Nash equilibrium exists is NP-complete, even for unweighted players. Recall that in our unweighted singleton game a NE is guaranteed to exist. Second, we show a tight upper bound on the price of anarchy for the sum of weighted costs.

4.1 Unweighted Games

In this subsection, we restrict ourselves to unweighted congestion games with priority lists, i.e., $w_i = 1$ for all $i \in N$. We first provide an example that shows that a Nash equilibrium need not exist. Farzad et al. [11] give a different example with two players for which a NE need not exist. Our example describes a symmetric game.

Example 2. The game, $G^\star$ contains 3 unweighted players, $w_i = 1$ for all $i \in N$, and 6 resources. Each players $i \in N$ has two pure strategies: $\{e_1, e_2, e_3\}$ and $\{e_4, e_5, e_6\}$. The delays are equal to 1 for all resources, and the priority lists are $\pi_j(i) = i + j - 1$ (modulo 3) for all $j \in E$ and $i \in N$. Observe that there is no Nash equilibrium if all three players choose the same three resources. Also, due to the Condorcet paradox [8], there is no Nash equilibrium in which two players choose one subset of resources and the other player chooses the other. Specifically, one of these two players has cost 5 and the other has cost 4. By deviating to the other triplet of resources, the player whose cost is 5 can reduce its cost to 4.

A natural question is to decide whether a game instance with unweighted players have a NE profile. Our next result shows that this is NP-complete. The hardness proof is different from the one in Theorem 1, since this proof considers unweighted players and multiple-resources strategies, while that proof is for weighted players and singleton strategies.

Theorem 13. *Given an instance of a congestion game with priority lists, it is NP-complete to decide whether the game has a NE profile. This is valid also for unweighted players.*

4.2 Equilibrium Inefficiency

We consider the sum of weighted players' costs as a measure of the quality of a strategy profile. Our analysis below is for linear cost functions, and is trivially extended to affine cost functions. A game G is said to be (λ, μ)-*smooth* if for all strategy profiles s, s' we have

$$\sum_{i \in N} cost_i(s'_i, s_{-i}) \leq \lambda \cdot cost(s') + \mu \cdot cost(s).$$

Roughgarden [22] showed that if a game G is (λ, μ)-smooth with $\lambda > 0$ and $\mu < 1$, then PoA(G)$\leq \frac{\lambda}{1-\mu}$.

Theorem 14. *Every congestion game with resource-specific priority lists is* $\left(2, \frac{1}{2}\right)$-*smooth. Hence* $PoA(\mathcal{G}) \leq 4$.

Proof. Given a strategy profile s, define $w_e(s) = \sum\limits_{i' \in N : e \in s_{i'}} w_{i'}$. For all s, s',

$$\begin{aligned}
&\sum_{i \in N} cost_i(s'_i, s_{-i}) \\
&\leq \sum_{i \in N} \sum_{e \in s'_i} w_i \cdot c_e \cdot (w_e(s) + w_i) = \sum_{e \in E} c_e \cdot \left(w_e(s') \cdot w_e(s) + \sum_{i \in N : e \in s'_i} w_i^2 \right) \\
&\leq \sum_{e \in E} c_e \cdot \left(w_e(s')^2 + \frac{1}{4} \cdot w_e(s)^2 + \sum_{i \in N : e \in s'_i} w_i^2 \right) \leq 2 \cdot cost(s') + \frac{1}{2} \cdot cost(s),
\end{aligned}$$

where the second inequality follows from $\left(w_e(s') - \frac{1}{2} \cdot w_e(s)\right)^2 \geq 0$ and the third inequality from $cost(s) = \sum_{e \in E} \frac{1}{2} \cdot c_e \cdot \left(w_e(s)^2 + \sum_{i \in N : e \in s_i} w_i^2\right)$ for all s. □

Correa and Queyranne [9] give an example that shows that the bound of 4 is tight for restricted singleton congestion games with priority lists derived from Smith's rule.

References

1. Ackermann, H., Goldberg, P., Mirrokni, V.S., Röglin, H., Vöcking, B.: A unified approach to congestion games and two-sided markets. Internet Mathematics **5**(4), 439–457 (2008)
2. Anshelevich, E., Dasgupta, A., Kleinberg, J., Tardos, E., Wexler, T., Roughgarden, T.: The price of stability for network design with fair cost allocation. SIAM Journal on Computing **38**(4), 1602–1623 (2008)

3. Awerbuch, B., Azar, Y., Richter, Y., Tsur, D.: Tradeoffs in worst-case equilibria. Theoretical Computer Science **361**(2-3), 200–209 (2006)
4. Caragiannis, I., Gkatzelis, V., Vinci, C.: Coordination mechanisms, cost-sharing, and approximation algorithms for scheduling. In: International Conference on Web and Internet Economics. pp. 74–87. Springer (2017)
5. Cho, Y., Sahni, S.: Bounds for list schedules on uniform processors. SIAM Journal on Computing **9**(1), 91–103 (1980). https://doi.org/10.1137/0209007
6. Christodoulou, G., Koutsoupias, E., Nanavati, A.: Coordination mechanisms. In: International Colloquium on Automata, Languages, and Programming. pp. 345–357. Springer (2004)
7. Cole, R., Correa, J., Gkatzelis, V., Mirrokni, V., Olver, N.: Decentralized utilitarian mechanisms for scheduling games. Games and Economic Behavior **92**, 306–326 (2015)
8. Marquis de Condorcet, M.J.A.: Essai sur l'application de l'analyse a la probabilite des decisions: rendues a la pluralite de voix. De l'Imprimerie royale (1785)
9. Correa, J., Queyranne, M.: Efficiency of equilibria in restricted uniform machine scheduling with total weighted completion time as social cost. Naval Research Logistics **59**(5), 384–395 (2012)
10. Czumaj, A., Vöcking, B.: Tight bounds for worst-case equilibria. ACM Transactions on Algorithms **3**(1), 4 (2007)
11. Farzad, B., Olver, N., Vetta, A.: A priority-based model of routing. Chicago Journal of Theoretical Computer Science **1** (2008)
12. Gairing, M., Lücking, T., Mavronicolas, M., Monien, B.: Computing nash equilibria for scheduling on restricted parallel links. Theory of Computing Systems **47**(2), 405–432 (2010)
13. Gourvès, L., Monnot, J., Moretti, S., Thang, N.K.: Congestion games with capacitated resources. Theory of Computing Systems **57**(3), 598–616 (2015)
14. Graham, R.: Bounds for certain multiprocessing anomalies. Bell System Technical Journal **45**(9), 1563–1581 (1966)
15. Hoeksma, R., Uetz, M.: The price of anarchy for utilitarian scheduling games on related machines. Discrete optimization **31**, 29–39 (2019)
16. Ibarra, O.H., Kim, C.E.: Heuristic algorithms for scheduling independent tasks on nonidentical processors. Journal of the ACM (JACM) **24**(2), 280–289 (1977)
17. Immorlica, N., Li, L.E., Mirrokni, V., Schulz, A.: Coordination mechanisms for selfish scheduling. Theoretical computer science **410**(17), 1589–1598 (2009)
18. Kann, V.: Maximum bounded 3-dimensional matching is max snp-complete. Information Processing Letters **37**(1), 27–35 (1991)
19. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibria. In: Annual Symposium on Theoretical Aspects of Computer Science. pp. 404–413. Springer (1999)
20. Piliouras, G., Nikolova, E., Shamma, J.S.: Risk sensitivity of price of anarchy under uncertainty. ACM Trans. Economics and Comput. **5**, 5:1–5:27 (2016)
21. Rosenthal, R.: A class of games possessing pure-strategy nash equilibria. International Journal of Game Theory **2**(1), 65–67 (1973)
22. Roughgarden, T.: Intrinsic robustness of the price of anarchy. Journal of the ACM **62**(5), 32 (2015)
23. Smith, W.E.: Various optimizers for single-stage production. Naval Research Logistics Quarterly **3**(1-2), 59–66 (1956)
24. Vöcking, B.: Algorithmic Game Theory, chap. 20: Selfish Load Balancing. Cambridge University Press (2007)