Coordination Mechanisms for Selfish Scheduling

Nicole Immorlica[1⋆], Li (Erran) Li[2], Vahab S. Mirrokni[1⋆⋆], and Andreas Schulz[3]

[1] Massachusetts Institute of Technology, Cambridge, MA, USA
[2] Center for Networking Research, Bell Labs, Murray Hill, NJ, USA,
[3] Operations Research Center, MIT, Cambridge, MA, USA,

Abstract. In machine scheduling, a set of n jobs must be scheduled on a set of m machines. Each job i incurs a processing time of p_{ij} on machine j and the goal is to schedule jobs so as to minimize some global objective function, such as the maximum makespan of the schedule considered in this paper. Often in practice, each job is controlled by an independent selfish agent who chooses to schedule his job on machine which minimizes the (expected) completion time of his job. This scenario can be formalized as a game in which the players are job owners; the strategies are machines; and the disutility to each player in a strategy profile is the completion time of his job in the corresponding schedule (a player's objective is to minimize his disutility). The equilibria of these games may result in larger-than-optimal overall makespan. The ratio of the worst-case equilibrium makespan to the optimal makespan is called the *price of anarchy* of the game. In this paper, we design and analyze scheduling policies, or *coordination mechanisms*, for machines which aim to minimize the price of anarchy (restricted to pure Nash equilibria) of the corresponding game. We study coordination mechanisms for four classes of multiprocessor machine scheduling problems and derive upper and lower bounds for the price of anarchy of these mechanisms. For several of the proposed mechanisms, we also are able to prove that the system converges to a pure Nash equilibrium in a linear number of rounds. Finally, we note that our results are applicable to several practical problems arising in networking.

1 Introduction

With the advent of the Internet, large-scale autonomous systems have become increasingly common. These systems consist of many independent and selfish agents all competing to share a common resource like bandwidth in a network or processing power in a parallel computing environment. Practical settings range from smart routing among stub autonomous systems (AS) in the Internet [14] to selfish user association in wireless local area networks [15]. In many systems of this kind, it is infeasible to impose some centralized control upon the users.

⋆ The final parts of this paper were completed while the author was at the Theory group in Microsoft Research.

⋆⋆ The final parts of this paper were completed while the author was at the Strategic Planning and Optimization group in amazon.com.

Rather, a centralized authority can only design protocols a priori and hope that the independent and selfish choices of the users—given the rules of the protocol—combine to create socially desirable results.

This approach, termed *mechanism design* has received considerable attention in the recent literature (see, for example, [22]). A common goal is to design system-wide rules which, given the selfish decisions of the users, maximize the total social welfare. The degree to which these rules approximate the social welfare in a worst-case equilibrium is known as the *price of anarchy* of the mechanism, and was introduced in 1999 by Koutsoupias and Papadimitriou [18] in the context of *selfish scheduling games* (see below for details). This seminal paper spawned a series of results which attempt to design mechanisms with minimum price of anarchy for a variety of games. One approach for achieving this goal [4, 7, 12] is to impose economic incentives upon users in the form of *tolls*; i.e., the disutility of a user is affected by a monetary payment to some central authority for the use of a particular strategy such as a route in a network. Another approach [3, 17, 23, 27] assumes that the central authority is able to enforce particular strategies upon some fraction of the users and thus perhaps utilize an unpopular resource. This action of the central authority is called a *Stackelberg strategy*.

A drawback of the above approaches is that many of the known algorithms assume global knowledge of the system and thus have high communication complexity. In many settings, it is important to be able to compute mechanisms locally. A third approach, which we follow here, is called *coordination mechanisms*, first introduced by Koutsoupias and Nanavati [6]. A coordination mechanism is a *local* policy that assigns a cost to each strategy s, where the cost of s is a function of the agents who have chosen s. Consider, for example, the *selfish scheduling game* in which there are n jobs owned by independent agents, m machines, and a processing time p_{ij} for job i on machine j. Each agent selects a machine on which to schedule its job with the objective of minimizing its own completion time. The social objective is to minimize the maximum completion time. A *coordination mechanism* [6] for this game is a local policy, one for each machine, that determines how to schedule jobs assigned to that machine. It is important to emphasize that a machine's policy is a function only of the jobs assigned to that machine. This allows the policy to be implemented in a completely distributed fashion.

Coordination mechanisms are closely related to local search algorithms. A local search algorithm iteratively selects a solution "close" to the current solution which improves the global objective. It selects the new solution from among those within some *search neighborhood* of the current solution. Given a coordination mechanism, we can define a local search algorithm whose search neighborhood is the set of best responses for each agent. Similarly, given a local search algorithm, it is sometimes possible to define a coordination mechanism whose pure strategy equilibria are local optima with respect to the search neighborhood. The locality gap of the search neighborhood, or approximation factor of the local search algorithm, is precisely the price of anarchy of the corresponding coordination mechanism and vice versa. In particular, designing new coordination mechanisms

may lead to the discovery of a new local search algorithms for a particular problem.

In this paper, we are primarily interested in the properties of *pure strategy Nash equilibria*[4] for the selfish scheduling games we define. A pure strategy Nash equilibrium is an assignment of jobs to machines such that no job has a unilateral incentive to switch to another machine. Although a non-cooperative game always has a mixed strategy equilibrium [21], it may in general not have a pure strategy equilibrium. However, by restricting ourselves to pure strategies, we are able to bound the *rate of convergence* of the mechanism. That is, if the jobs, starting from an arbitrary solution, iteratively play their best response strategies, how long does it take for the mechanism to reach a pure Nash equilibrium? Guarantees of this sort are important for bounded price-of-anarchy mechanisms to be applicable in a practical setting.

Preliminaries. In machine scheduling, there are n jobs must be processed on m machines. Job i has processing time p_{ij} on machine j. A schedule μ is a function mapping each job to a machine. The makespan of a machine j in schedule μ is $M_j = \sum_{i:j=\mu(i)} p_{ij}$. The goal is to find a schedule μ which minimizes the maximum makespan, $C_{\max} = \max_j M_j$. Different assumptions regarding the relationship between processing times yield different scheduling problems, of which we consider the following four: (i) *Identical* machine scheduling ($P||C_{\max}$) in which $p_{ij} = p_{ik} = p_i$ for each job i and machines j and k; (ii) *Uniform* or *related* machine scheduling ($Q||C_{\max}$) in which $p_{ij} = \frac{p_i}{s_j}$, where p_i is the load of job i and $s_j \leq 1$ is the speed of machine j; (iii) machine scheduling for *restricted assignment* or *bipartite* machine scheduling ($B||C_{\max}$) in which each job i can be scheduled on a subset S_i of machines, i.e., p_{ij} is equal to p_i if $j \in S_i$ and is equal to ∞ otherwise; and (iv) *unrelated* machine scheduling ($R||C_{\max}$) in which the processing times p_{ij} are arbitrary positive numbers.

In *selfish scheduling*, each job is owned by an independent agent whose goal is to minimize the completion time of his own job. To induce these selfish agents to take globally near-optimal actions, we introduce the notion of a *coordination mechanism* [6]. A coordination mechanism is a set of *scheduling policies*, one for each machine. A scheduling policy for a machine j takes as input a set $S \subseteq \{p_{1j}, \ldots, p_{nj}\}$ of jobs on machine j along with their processing times on machine j and outputs an ordering in which they will be scheduled. The policy is run locally at a machine, and so does not have access to information regarding the global state of the system (the set of all jobs, for instance, or the processing times of the jobs on other machines). A coordination mechanism defines a game in which there are n agents. An agent's strategy set is the set of possible machines $\{1, ..., m\}$. Given a strategy profile, the disutility of agent i is the (expected) completion time of job i in the schedule defined by the coordination mechanism. We study four coordination mechanisms. In the ShortestFirst and LongestFirst policies, we sequence the jobs in non-decreasing and non-increasing order of their processing times, respectively. In the Randomized policy, we process the

[4] In this paper, we use the term Nash equilibria for pure Nash equilibria.

	Makespan	ShortestFirst	LongestFirst	Randomized
$P\|\|C_{\max}$	$2-\frac{2}{m+1}$ [11, 28]	$2-\frac{2}{m+1}$ [11, 28]	$\frac{4}{3}-\frac{1}{3m}$ [6]	$2-\frac{2}{m}$ [11, 28]
$Q\|\|C_{\max}$	$\Theta(\frac{\log m}{\log\log m})$ [8]	$\Theta(\log m)$ ([1], *)	$\frac{4}{3}-\frac{1}{3m} \le P \le 2-\frac{2}{m}$ *	$\Theta(\frac{\log m}{\log\log m})$([8])
$B\|\|C_{\max}$	$\Theta(\frac{\log m}{\log\log m})$ [13]	$\Theta(\log m)$ ([1], *)	$\Theta(\log m)$ ([2], *)	$\Theta(\frac{\log m}{\log\log m})$ [13]
$R\|\|C_{\max}$	Unbounded [28]	[9] $\log m \le P \le m$ *	Unbounded	$\Theta(m)$ *

Table 1. The price of anarchy for four different policies and scheduling problems. All the upper and lower bounds hold for pure Nash equilibria. The upper bounds of the Randomized policy for $R||C_{\max}$ and $Q||C_{\max}$ are valid for the maximum of the expected load on any machine in mixed Nash equilibria. The results marked by * are proved in this paper.

jobs in a random order.[5] In the Makespan policy, we process all jobs on the same machine in parallel, and so the completion time of a job on machine j is the makespan of machine j.

We are interested in the properties of the solution imposed by the selfish strategies of the agents in a pure Nash equilibrium, in particular, the *price of anarchy*. In this setting, the price of anarchy is the worst-case ratio (over all instances) of the maximum makespan in a (pure) Nash equilibrium to the optimal makespan. For each policy and each scheduling problem, we prove upper and lower bounds on the price of anarchy. Some of these bounds are already known as the approximation factor of some local search algorithms or the price of anarchy in some selfish load balancing games. All bounds, known and new, are summarized in Table 1.

Related work. The Makespan policy [8, 13, 18] is perhaps the best known policy among the above policies. Czumaj and Vocking [8] give tight results on the price of anarchy for the Makespan policy for mixed Nash equilibria and $Q||C_{\max}$. Gairing et al. [13] study the Makespan policy for $B||C_{\max}$ and give a polynomial-time algorithm for computing a pure Nash equilibrium with makespan at most twice the optimal makespan. They also give a tight bound for the price of anarchy of the Makespan policy for pure Nash equilibria and $B||C_{\max}$. Azar et al. [1, 2] proved that the greedy list scheduling algorithm is an $O(\log m)$-approximation algorithm for $Q||C_{\max}$ and $B||C_{\max}$. Their proof can be used to bound the price of anarchy of the LongestFirst and ShortestFirst polices for $B||C_{\max}$ and $Q||C_{\max}$.

Coordination mechanism design was introduced by Koutsoupias and Nanavati [6]. In their paper, they analyzed the LongestFirst policy for $P||C_{\max}$ and also studied a selfish routing game. As we have mentioned before, the price of anarchy for coordination mechanisms is closely related to the approximation factor of local search algorithms. The speed of convergence and the approximation factor of local search algorithms for scheduling problems were studied in several papers [9–11, 16, 24, 25, 28]. Vredeveld surveyed some of the results on local

[5] The Randomized policy is also known as the batch model [18].

search algorithms for scheduling problems in his thesis [28]. The jump model in his survey [28] is similar to the Makespan policy. Moreover, the push model in his survey is related to the LongestFirst policy. It is known that the jump local search algorithm is an $\Theta(\sqrt{m})$-approximation for $Q||C_{\max}$ [24, 28]. Cho and Sahni [24] showed that the approximation factor of the shortest-first greedy algorithm is not better than $\log m$ for $Q||C_{\max}$.

Ibarra and Kim [16] analyzed several greedy algorithms for $R||C_{\max}$. In particular, they proved that the shortest-first greedy algorithm is an m-approximation for $R||C_{\max}$. Davis and Jaffe [9] showed that the approximation factor of this greedy algorithm is at least $\log m$. Our lower bound example for the ShortestFirst and the LongestFirst policy is the same as the example in [9]. Davis and Jaffe [9] also gave a $\sqrt{m}$-approximation for $R||C_{\max}$. The best known approximation factor for $R||C_{\max}$ is given by a 2-approximation algorithm due to Lenstra, Shmoys and Tardos [19].

Even-Dar et al. [10] considered the convergence time to Nash equilibria for variants of the selfish scheduling problem. In particular, they studied the Makespan policy and bounded the number of required steps to reach a pure Nash equilibrium.

Our Contribution. We give almost tight bounds for the price of anarchy of pure Nash equilibria for the Randomized, the ShortestFirst, and the LongestFirst policies. We give a proof that the price of anarchy of any deterministic coordination mechanism (including ShortestFirst and LongestFirst) for $Q||C_{\max}$ and $B||C_{\max}$ is at most $O(\log m)$. This result is also implied in the results of Azar et al [1, 2] on the approximation factor of the greedy list scheduling algorithm for $Q||C_{\max}$ and $B||C_{\max}$. We also prove that any coordination mechanism based on a *universal ordering* (such as ShortestFirst and LongestFirst) has a price of anarchy $\Omega(\log m)$ for $B||C_{\max}$. Moreover, we show that the price of anarchy of the LongestFirst policy for $Q||C_{\max}$ is at most $2 - \frac{2}{m}$. In addition, we analyze the Randomized policy for $R||C_{\max}$ machine scheduling and prove a bound of $\Theta(m)$ for the price of anarchy of this policy. We further study the convergence and existence of pure Nash equilibria for the ShortestFirst and LongestFirst policies. In particular, we show that the mechanism based on the ShortestFirst policy is a potential game, and thus any sequence of best responses converges to a Nash equilibrium after at most n rounds. We also prove fast convergence of the LongestFirst policy for $Q||C_{\max}$ and $B||C_{\max}$.

2 Upper Bounds on the Price of Anarchy

2.1 The ShortestFirst Policy

We begin by bounding the price of anarchy of the ShortestFirst policy for $R||C_{\max}$. We note that there is a direct correspondence between outputs of the well-known shortest-first greedy algorithm for machine scheduling (see [16], "Algorithm D") and pure Nash equilibria of the ShortestFirst policy in $R||C_{\max}$.

Theorem 1. *The set of Nash equilibria for the* ShortestFirst *policy in* $R||C_{\max}$ *is precisely the set of solutions that can be output by the shortest-first greedy algorithm.*

The proof of this theorem is deferred to the full version of this paper. The implication is that any bound on the approximation factor of the shortest-first greedy algorithm is also a bound on the price of anarchy of the ShortestFirst policy for $R||C_{\max}$. In particular, we may use this theorem and the result of Ibarra and Kim [16] to prove that the price of anarchy of ShortestFirst for $R||C_{\max}$ is at most m.

Theorem 2. *The price of anarchy of the* ShortestFirst *policy for* $R||C_{max}$ *is at most* m.

We can further prove that the price of anarchy of the ShortestFirst policy for $Q||C_{\max}$ is $\Theta(\log m)$. This also shows that the approximation factor of the shortest-first greedy algorithm for $Q||C_{\max}$ is $\Theta(\log m)$, as was previously observed by Azar et al [1]. In fact, the bound on the price of anarchy can be derived from their result as well. Our proof, derived independently, uses ideas from the proof of the price of anarchy for the Makespan policy for $Q||C_{\max}$ by Czumaj and Vocking [8].

We prove our upper bound for any *deterministic* coordination mechanism. A coordination mechanism is deterministic if the scheduling policies do not use randomization to determine the schedules. We prove that the price of anarchy for deterministic mechanisms, and in particular for the ShortestFirst policy, for $Q||C_{\max}$ is at most $O(\log m)$. The proof is deferred to the full version of the paper.

Theorem 3. *The price of anarchy of a deterministic policy for* $Q||C_{\max}$ *is at most* $O(\log m)$. *In particular, the price of anarchy of the* ShortestFirst *policy for* $Q||C_{\max}$ *is* $O(\log m)$.

In addition, we can prove that the price of anarchy of any deterministic mechanism for $B||C_{\max}$ is $\Theta(\log m)$. This result is implied by a result of Azar et al [2] on the approximation factor of the greedy algorithm for $B||C_{\max}$, but our proof is independent of theirs and uses the ideas of the proof for the Makespan policy by Gairing et al. [13]. We defer the proof to the full version of the paper.

Theorem 4. *The price of anarchy of a deterministic policy for* $B||C_{\max}$ *is at most* $O(\log m)$. *In particular the price of anarchy of the* ShortestFirst *policy for* $B||C_{\max}$ *is* $O(\log m)$.

2.2 The LongestFirst Policy

It is easy to see that the price of anarchy of the LongestFirst policy for unrelated machine scheduling is unbounded. It is known that the price of anarchy of this policy for $P||C_{\max}$ is bounded above by $\frac{4}{3} - \frac{1}{3m}$ [6]. Theorems 4 and 3 show

that the price of anarchy of the LongestFirst policy for $B||C_{\max}$ and $Q||C_{\max}$ is at most $O(\log m)$. In Section 3, we prove that this bound is tight for $B||C_{\max}$. Here, we prove that the price of anarchy of the LongestFirst policy for $Q||C_{\max}$ is at most $2 - \frac{2}{m}$. Note that this is the only policy for which we know that the price of anarchy for $Q||C_{\max}$ is bounded by a constant. The proof is similar to the proof of Theorem 4.9 in [28], and is deferred to the full version of the paper.

Theorem 5. *The price of anarchy of the* LongestFirst *policy for related machine scheduling ($Q||C_{\max}$) is at most* $2 - \frac{2}{m}$.

2.3 The Randomized Policy

In the Randomized policy, an agent's disutility is the *expected* completion time of his job. We begin by computing the condition under which an agent has an incentive to change strategies. Consider a job i on machine j and let J_j be the set of jobs assigned to machine j. Then the disutility of agent i under the Randomized policy is:

$$p_{ij} + \frac{1}{2} \sum_{i' \neq i, i' \in J_j} p_{i'j}.$$

Letting M_j be the makespan of machine j, we see that a job i on machine j has an incentive to change to machine k if and only if:

$$p_{ij} + M_j > 2p_{ik} + M_k.$$

Because of this observation, the randomized policy is the same as the Makespan policy for $P||C_{\max}$ and $B||C_{\max}$. This implies a price of anarchy of at most $2 - \frac{2}{m}$ and $O(\frac{\log m}{\log \log m})$ for these settings, respectively.

Here, we bound the price of anarchy of the Randomized policy for $Q||C_{\max}$ and $R||C_{\max}$. In fact, we prove that, in contrast to the Makespan policy the price of anarchy of the Randomized policy for $R||C_{\max}$ is not unbounded.

Theorem 6. *The price of anarchy of the* Randomized *policy for $R||C_{\max}$ is at most* $2m - 1$.

Proof. Let $\mathcal{L}$ be any pure strategy Nash equilibrium and $\mathcal{O}$ be an optimal solution. We consider two groups of jobs — those that are on different machines in $\mathcal{O}$ and $\mathcal{L}$, and those that are on the same machine. Define S_{qj} as the set of jobs on machine q in $\mathcal{L}$ that are on machine j in $\mathcal{O}$, and let $L_q = \sum_{i \in \cup_{j \neq q} S_{qj}} p_{iq}$, $O_q = \sum_{i \in \cup_{j \neq q} S_{jq}} p_{iq}$, and $R_q = \sum_{i \in S_{qq}} p_{iq}$. Thus, the makespan of a machine l in $\mathcal{L}$ is $L_l + R_l$, and the makespan of l in $\mathcal{O}$ is $O_l + R_l$. Since $\mathcal{L}$ is a Nash equilibrium, for all jobs $i \in S_{qj}$,

$$L_q + R_q + p_{iq} \leq L_j + R_j + 2p_{ij}.$$

Suppose the makespan of $\mathcal{L}$ is achieved on machine l and the makespan of $\mathcal{O}$ is achieved on machine l'. Then

$$
\begin{aligned}
|\cup_{j\neq l} S_{lj}|(L_l+R_l)+L_l &= \sum_{i\in\cup_{j\neq l}S_{lj}} (L_l+R_l+p_{il}) \\
&\leq \sum_{i\in\cup_{j\neq l}S_{lj}} (L_j+R_j+2p_{ij}) \\
&\leq |\cup_{j\neq l} S_{lj}|(L_l+R_l)+2\sum_{j\neq l}\sum_{i\in S_{lj}} p_{ij} \\
&\leq |\cup_{j\neq l} S_{lj}|(L_l+R_l)+2\sum_{j\neq l} O_j \\
&\leq |\cup_{j\neq l} S_{lj}|(L_l+R_l)+2(m-1)(O_{l'}+R_{l'}). \quad (1)
\end{aligned}
$$

Therefore, the value of the solution induced by the Nash equilibrium $\mathcal{L}$ is at most $2(m-1)(O_{l'}+R_{l'})+R_l \leq (2m-1)(O_{l'}+R_{l'})$, and so the price of anarchy is at most $2m-1$.

Unfortunately, we do not know if pure Nash equilibria exist for the Randomized policy for $R||C_{\max}$, and so the above theorem might be vacuous. However, we can extend the above proof to bound the maximum of the expected load of a machine in a mixed Nash equilibrium of the Randomized policy for $R||C_{\max}$. If M_j is the expected load of machine j in a mixed Nash equilibrium, then it is easy to show that if the probability of assigning job i to machine q is nonzero, then for any other machine j, $M_q + p_{iq} \leq M_j + 2p_{ij}$. Now, we can define L_q as the expected load of jobs with positive probability on machine q that are scheduled on machines other than q in the optimum solution. Similar inequalities hold in this setting. This bounds the maximum of the expected load of any machine in a mixed Nash equilibrium. We defer the details of the proof to the full version of the paper. Note that this analysis does not hold for the expected value of the maximum load (see [8] for the difference between these two objective functions). We can further prove that this bound is tight, up to a constant factor (see Theorem 9).

Finally, we also observe a better bound for $Q||C_{\max}$. The proof is along the same lines as the proof of the Makespan policy [8] and is omitted here. This proof is also valid for the maximum expected load on any machine for mixed Nash equilibria.

Theorem 7. *The price of anarchy of the* Randomized *policy for* $Q||C_{\max}$ *is at most* $O(\frac{\log m}{\log\log m})$.

3 Lower Bounds on the Price of Anarchy

In this section, we prove lower bounds on the price of anarchy of coordination mechanisms. Our first result shows that the price of anarchy of a general class of coordination mechanisms for $B||C_{\max}$ and $R||C_{\max}$ is at least $\log m$. This is interesting in light of the fact that constant-factor LP-based approximation

algorithms are known for $R||C_{\max}$ [19], and suggests that it may be hard to obtain similar approximations with local search algorithms.

We consider a class of mechanisms which are *deterministic* and use a *common tie-breaking rule*. That is, there is no randomization in the scheduling policies; and, whenever two jobs i and i' have the same processing time on a machine, one of them, say i, is *always* scheduled before the other (independent of the machine and the presence of other jobs). An example of such a mechanism is ShortestFirst with an alphabetical tie-breaking rule (i.e. if $p_{ij} = p_{i'j}$ then i is scheduled before i' if and only if $i < i'$). The example in our proof was used by Davis and Jaffe [9] to show that the approximation factor of the shortest-first greedy algorithm is at least $\log m$.

Theorem 8. *The price of anarchy of any deterministic coordination mechanism which uses a common tie-breaking rule is at least* $\log_2 m$ *for* $B||C_{\max}$ *and* $R||C_{\max}$.

Proof. Consider a deterministic coordination mechanism with a common tie-breaking rule, say alphabetically first (this assumption is without-loss-of-generality since we can always relabel jobs for the purposes of the proof such that in a tie job i is scheduled before i' whenever $i < i'$). Consider the following instance of $B||C_{\max}$: there are m jobs and m machines and the processing time of job i on machines $1, 2, \ldots, m - i + 1$ is 1. Job i cannot be scheduled on machines $m - i + 2, \ldots, m$. Assume that m is a power of 2 and $m = 2^k$.

Consider an assignment of jobs to machines as follows. Jobs 1 to $2^{k-1} = \frac{m}{2}$ are assigned to machines 1 to $\frac{m}{2}$ respectively. Jobs $\frac{m}{2} + 1$ to $\frac{3m}{4}$ are assigned to machines 1 to $\frac{m}{4}$ respectively. Jobs $\frac{3m}{4} + 1$ to $\frac{7m}{8}$ are assigned to machines 1 to $\frac{m}{8}$ respectively, and so on. It is not hard to check that this is a pure strategy Nash equilibrium of this mechanism. The makespan of this assignment is $k = \log_2 m$. In the optimal assignment job i is assigned to machine $m - i + 1$. Thus the optimal makespan is 1, and so the price of anarchy is at least $\log_2 m$.

The example in the above proof can be easily changed to show that for $R||C_{\max}$, the price of anarchy of the LongestFirst and ShortestFirst policies is at least $\log_2 m$, even if there is no tie among the processing times.

Theorem 8 proves that if a coordination mechanism is deterministic and policies of different machines are the same, then we cannot hope to get a factor better than $\log_2 m$ for $R||C_{\max}$. One might hope that the Randomized policy can achieve a constant price of anarchy. However, we have the following lower bound for the Randomized policy.

Theorem 9. *The price of anarchy of the* Randomized *policy for* $R||C_{\max}$ *is at least* $m - 1$.

Proof. Consider a scenario with m machines and $(m-1)^2$ jobs. Split the first $(m-1)(m-2)$ jobs into $m-1$ groups $J_1, \ldots, J_{m-1}$, each of size $m-2$ jobs. For jobs $i \in J_k$, let $p_{ik} = 1$, $p_{im} = 1/m^2$, and $p_{ij} = \infty$ for all other machines j. Form a matching between the remaining $m - 1$ jobs and the first $m - 1$ machines.

Whenever job i is matched to machine j in this matching, set $p_{ij} = 1$ and $p_{im} = 1$.

The optimal solution has makespan 1 and assigns all jobs in $J_1, \ldots, J_{m-1}$ to the last machine and each of the remaining $m-1$ jobs to its corresponding machine in the matching. However, the solution which assigns all jobs in J_k to machine k for all $1 \leq k \leq m-1$ and all remaining $m-1$ jobs to machine m is a Nash equilibrium with makespan $m-1$. To see that this is a pure strategy Nash equilibrium, consider a job $i \in J_k$. Its disutility on machine k is $\frac{1}{2}(m-3)+1$ while its disutility if it moved to machine m would increase to $\frac{1}{2}(m-1)+1/m^2$. Therefore all jobs in $J_1, \ldots, J_{m-1}$ are playing a best response to the current set of strategies. Now consider one of the remaining $m-1$ jobs i. Say job i is matched to machine j in the matching. Then the disutility of job i on machine m is $\frac{1}{2}(m-2)+1$ while its disutility if it moved to machine j would remain $\frac{1}{2}(m-2)+1$. Since these jobs are also playing a (weakly) best response to the current set of strategies, the above scenario is a Nash equilibrium in the Randomized policy.

4 Convergence to Pure Strategy Nash Equilibria

In practice, it is undesirable if the job to machine mapping keeps changing. The system performance can be adversely affected if players keep reacting to one another's changes of strategies. A good coordination mechanism is one with a small price of anarchy and fast convergence to pure strategy Nash equilibrium. In this section we investigate the convergence of players' selfish behavior. We prove that, except for the case of the Randomized and LongestFirst policies for $R||C_{\max}$ and the Randomized policy for $Q||C_{\max}$, the selfish behavior of players converges to a pure Nash equilibrium.

We first define the notion of a state graph and a potential function. Let A_i be the set of actions of player i, $i = 1, 2, \cdots, n$. In our setting, each A_i equals the set of machines $\{1, \ldots, \}$. A state graph $G = (V, E)$ is a directed graph where V is the set of nodes $A_1 \times A_2 \cdots A_n$, and an edge labelled with i exists from state u to v if the only difference between the two states is the action of player i and i's payoff is strictly less in v. A pure strategy Nash equilibrium corresponds to a node with no outgoing edges. A potential function is a function f mapping the set of states to a totally ordered set such that $f(v)$ is strictly less than $f(u)$ for all edges $uv \in G$. In other words, whenever a player in state u changes his action and improves his payoff, the resulting state v satisfies $f(u) > f(v)$. Note that the existence of a potential function implies the state graph is acyclic and establishes the existence of pure strategy Nash equilibrium. The existence of a potential function also implies that the Nash dynamics will converge if one player takes the best response action atomically. We restrict our convergence analysis to this atomic and best-response behavior of players. A game that has a potential function is called a potential game. Many of our proofs proceed by showing that the games we have defined in this paper are in fact potential games.

We remark that the Makespan policy corresponds to a potential game. This fact has been observed in various places. In particular, Even-Dar et al. [10] give several bounds on the speed of convergence to pure Nash equilibria for this policy. The Randomized policy is the same as the Makespan policy for $B||C_{\max}$ and $P||C_{\max}$. Thus, the Randomized policy also corresponds to a potential game for $B||C_{\max}$ and $P||C_{\max}$. We do not know if the Randomized policy for $R||C_{\max}$ is a potential game or not. If it were, this would imply, among other things, that pure Nash equilibrium exist. For the rest of this section, we study the convergence for the ShortestFirst and LongestFirst policies.

4.1 Convergence for the ShortestFirst Policy.

In Section 2.1, we have shown that pure strategy Nash equilibria exist for the ShortestFirst policy for $R||C_{\max}$ and can be found in polynomial time. In the following, we show that this game is a potential game and players will converge to a pure strategy Nash equilibrium. Note that this gives an alternative proof of the existence of pure strategy Nash equilibria.

Theorem 10. *The* ShortestFirst *policy for* $R||C_{\max}$ *is a potential game.*

Proof. For any state u, let $c(u)$ be the vector of job completion times sorted in increasing order. We show that as a job switches from one machine to another machine to decrease its completion time, it decreases the corresponding vector c lexicographically. Suppose the system is in state u and $c(u) = (c_1, c_2, \ldots, c_n)$. Suppose job i with completion time c_i switches machines and decreases its completion time. Call the new state v and let $c(v) = (c'_1, c'_2, \ldots, c'_n)$. Let i's completion time in v be c'_j. We know that $c'_j < c_i$. However, the change in i's action may cause an increase in the completion times of other jobs. Assume that job i switched to machine k in state v. Jobs whose completion time increases after this switch are the jobs that are scheduled on machine k and whose processing time on machine k is greater than i's processing time on machine k. Thus, the completion times of these jobs in state u (before i moves) were greater than or equal to c'_j. Thus in the resulting vector $c(v)$, we decrease an element of the vector from c_i to c'_j and we do not increase any element with value less than c'_j. Thus this switch decreases the corresponding vectors lexicographically, i.e. $c(v) < c(u)$ and so c is a potential function. This completes the proof.

Corollary 1. *Selfish behavior of players will converge to a Nash equilibrium under the* ShortestFirst *policy for* $R||C_{\max}$.

Knowing that the selfish behavior of players converges to a Nash equilibrium and the social value of a Nash equilibrium is bounded does not indicate a fast convergence to good solutions. We are interested in the speed of convergence to a Nash equilibrium. We consider the best responses of jobs and prove fast convergence to Nash equilibria for the ShortestFirst policy.

In order to prove a convergence result, we must make some assumption regarding the manner in which the ShortestFirst policy resolves ties. In particular,

we will require that this tie-breaking rule is deterministic and satisfies *independence of irrelevant alternatives*, i.e., the resolution of a tie between jobs i and j is not affected by the presence or absence of job k. One such tie-breaking rule is the *alphabetically first* rule. When there is a tie between the processing times of two jobs, the alphabetically first rule always chooses the one with the smaller identifier. We note that all our upper and lower bounds hold for the ShortestFirst policy with the alphabetically first rule. For simplicity, in the proof below, we will assume our ShortestFirst policy employs the alphabetically first rule to break ties.

Theorem 11. *In $R||C_{\max}$ with the* ShortestFirst *policy, best responses of jobs converge to a Nash equilibrium after n rounds of any arbitrary ordering of jobs, when ties in the coordination mechanism are resolved using the alphabetically first rule. In other words, from any state in the state graph G, it takes at most n state traversals to end up in a state with no outgoing edges.*

Proof. In the t'th round, let i_t be the alphabetically first job which achieves the minimum possible disutility among the set of jobs $J_t \equiv J - \{i_1, \ldots, i_{t-1}\}$, fixing the strategies of jobs $J - \{i_t\}$. We prove by induction that in round t, job i_t moves to some machine and remains there in subsequent rounds.

Suppose j is any machine on which i_t achieves his minimum disutility. Then in the t'th round, a best response for i_t is to move to machine j. We show that this machine is the weakly best response of i_t for *any* set of strategies of jobs J_t. By weakly best response, we mean there is no other action that, gives the player a strictly better payoff (a smaller completion time in our setting).

First notice that the disutility of i_t on j can not increase as jobs in J_t alter their strategies. This is because any job $i' \in J_t$ has processing time at least $p_{i_t j}$ on machine j and, upon equality, is alphabetically larger or else we would have set $i_t = i'$ in the t'th round. Now consider some other machine j'. Let c_j be the completion time of job i on machine j. Then any job with completion time less than c_j on machine j' in round t must be in $\{i_1, \ldots, i_{t-1}\}$ or else we would have picked this job to be i_t in round t. Thus, the strategies of these jobs are fixed. Let i' be the job on machine j' in round t with the smallest completion time that is at least c_j. If $p_{i_t j'} < p_{i' j'}$, then the strategy of i' and all other jobs scheduled after i' on j' in round t does not affect i_t's disutility for machine j'. If $p_{i_t j'} \geq p_{i' j'}$, then even if i' leaves j' in a subsequent round, the completion time of i_t on j' is still at least i''s completion time on j' in round t, or at least c_j. Thus, it is a weakly best response for i_t to remain on machine j.

This shows that it is a weakly best response for i_t to remain on machine j in all subsequent rounds.

The next theorem proves that the bound of Theorem 11 is tight.

Theorem 12. *There are instances of $R||C_{\max}$ under the* ShortestFirst *policy for which it takes n rounds of best responses of players to converge to a Nash equilibrium.*

Proof. Suppose the processing time of job j on machine i is $1 + i\epsilon + (n-j)\frac{\epsilon}{n}$ for a sufficiently small ϵ. Starting from an empty assignment, let jobs go to their best machine in the order of $1, 2, \ldots, n$. In the first round, all jobs go to the first machine. In the second round, all jobs except the n'th job go to the second machine. In the i'th round, jobs $1, 2, n-i+1$ will go from machine $i-1$ to machine i. At the end, job i is scheduled on machine $n-i+1$ and it takes n rounds to converge to this equilibrium.

4.2 Convergence for the LongestFirst Policy.

In the LongestFirst policy, it is possible to prove convergence in the $Q||C_{\max}$, $B||C_{\max}$, and $P||C_{\max}$ models in a manner similar to that of Theorem 11. One just must argue that in each round the job with the longest feasible processing time (among jobs not yet considered) moves to its optimal machine and remains there in subsequent rounds. For the sake of brevity, we omit this proof.

Theorem 13. *For the* LongestFirst *policy in $Q||C_{\max}$, $B||C_{\max}$, or $P||C_{\max}$, the best responses of jobs converge to a Nash equilibrium after n rounds of any arbitrary ordering of jobs, when ties in the coordination mechanism are resolved using the alphabetically first rule.*

We note that Theorem 13 proves the existence of a Nash equilibrium in these games. We do not know how to prove that the LongestFirst policy converges in the $R||C_{\max}$model, although given that the price of anarchy would be unbounded anyway, such a proof is of somewhat questionable value.

5 Conclusion and Future Work

We have studied abstract scheduling games where the disutility of each player is its completion time. We note that our results can be applied in many practical network settings. Our results can be directly applied to the Internet setting [26] where there is a set of selfish clients, each of whom must choose a server from a set of servers. Each client tries to minimize its latency, or job completion time; the social welfare is the total system-wide latency. Similar problems arise in the wireless network setting. For example, the basic fairness and load balancing problem in wireless LANs [15] is reduced to the problem of unrelated parallel machine scheduling. Centralized algorithms have been designed for this problem in [15]. We hope to use ideas from our coordination mechanisms to design decentralized algorithms for this problem. In the third generation wireless data networks, the channel quality of a client is typically time-varying [5]; it would be interesting to study coordination mechanisms in this context given that users may exhibit selfish behavior.

Theoretically, the most interesting open problem in this paper is to find coordination mechanisms with constant price of anarchy for $B||C_{\max}$ and $R||C_{\max}$. We have shown that this cannot be achieved by any deterministic coordination mechanism which uses a common tie-breaking rule. Another problem left open

in this paper is the existence of pure Nash equilibria for the Randomized policy for $R||C_{\max}$.

Through-out this paper, we assumed all information regarding job processing times was public knowledge. A new direction considered in [6] and [20] is the design of coordination mechanisms in a private information setting, i.e. where a job's processing time is a private value. In such a setting, it would be nice if a coordination mechanism incentivizes jobs to announce their true processing times. The only constant-factor price of anarchy for $Q||C_{\max}$ and the best factor for $P||C_{\max}$ in our paper are achieved using the LongestFirst policy, but this policy is not truthful. In particular, jobs can artificially inflate their length (by inserting empty cycles, perhaps) and as a result actually decrease their disutility. A truthful coordination mechanism with a constant price of anarchy in these settings would be an interesting result.

References

1. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *J. ACM 44*, 3, 1997.
2. Y. Azar, J. Naor, and R. Rom. The competitiveness of on-line assignments. *Journal of Algorithms*, 18:221–237, 1995.
3. A. Bagchi. Stackelberg differential games in economic models. *Springer-Verlag*, 1984.
4. M. Beckman, C. B. McGuire, and C. B. Winsten. *Studies in the Economics of Transportation*. Yale University Press, 1956.
5. Sem Borst. User-level performance of channel-aware scheduling algorithms in wireless data networks. *IEEE/ACM Transaction on Networking*, 13(3):636–647, 2005.
6. G. Christodoulou, E. Koutsoupias, and A. Nanavati. Coordination mechanisms. pages 345–357, Turku, Finland, 12–16 July 2004.
7. R. Cole, Y. Dodis, and T. Roughgarden. How much can taxes help selfish routing? In *EC*, pages 98–107, 2003.
8. A. Czumaj and B. Vocking. Tight bounds for worst-case equilibria. In *SODA*, pages 413–420, 2002.
9. E. Davis and J.M. Jaffe. Algorithms for scheduling tasks on unrelated processors. *J. ACM*, 28(4):721–736, 1981.
10. E. Even-dar, A. Kesselman, and Y. Mansour. Convergence time to nash equilibria. In *ICALP*, pages 502–513, 2003.
11. G. Finn and E. Horowitz. A linear time approximation algorithm for multiprocessor scheduling. *BIT*, 19:312–320, 1979.
12. L. Fleischer, K. Jain, and M. Mahdian. Tolls for heterogeneous selfish users in multicommodity networks and generalized congestion games. In *FOCS*, pages 277–285, 2004.
13. M. Gairing, T. Lucking, M. Mavronicolas, and B. Monien. Computing nash equilibria for scheduling on restricted parallel links. In *STOC*, pages 613–622, 2004.
14. D. K. Goldenberg, L. Qiu, H. Xie, Y. R. Yang, and Y. Zhang. Optimizing cost and performance for multihoming. *SIGCOMM Comput. Commun. Rev.*, 34(4):79–92, 2004.
15. Y. Bejerano S.J. Han and L. Li. Fairness and load balancing in wireless LANs. In *Proceedings of the 9th annual international conference on Mobile computing and networking (MOBICOM)*, 2004.

16. O.H. Ibarra and C.E. Kim. Heuristic algorithms for scheduling independent tasks on nonidentical processors. *J. ACM*, 24(2):280–289, 1977.
17. Y.A. Korilis, A.A. Lazar, and A. Orda. Achieving network optima using Stackelberg routing strategies. *IEEE/ACM Transactions on Networking*, 5(1):161–173, 1997.
18. E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *STACS*, pages 404–413, 1999.
19. J. Lenstra, D. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.
20. H. Moulin. On scheduling fees to prevent merging, splitting and transferring of jobs. May 2004.
21. J. F. Nash. Equilibrium points in N-person games. In *Proceedings of NAS*, 1950.
22. N. Nisan and A. Ronen. Algorithmic mechanism design. In *STOC*, pages 129–140, 1999.
23. T. Roughgarden. Stackelberg scheduling strategies. In *STOC*, pages 104–113, 2001.
24. S. Sahni and Y. Cho. Bounds for list schedules on uniform processors. *Siam J. of Computing*, 9:91–103, 1980.
25. P. Schuurman and T. Vredeveld. Performance guarantees of local search for multiprocessor scheduling. In *IPCO*, pages 370–382, 2001.
26. S. Suri, C. D. Toth, and Y. Zhou. Selfish load balancing and atomic congestion games. In *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures(SPAA)*, pages 188–195, 2004.
27. H. von Stackelberg. Marktform und Gleichgewicht. *Springer-Verlag*, 1934. English translation entitled *The Theory of the Market Economy*.
28. T. Vredeveld. *Combinatorial approximation algorithms. Guaranteed versus experimental performance*. 2002. Ph.D. thesis.