Preemptive Coordination Mechanisms for Unrelated Machines[⋆]

Fidaa Abed[1] and Chien-Chung Huang[2]

[1] Max-Planck-Institut für Informatik. fabed@mpi-inf.mpg.de
[2] Humboldt-Universität zu Berlin. villars@informatik.hu-berlin.de

Abstract. We investigate coordination mechanisms that schedule n jobs on m unrelated machines. The objective is to minimize the latest completion of all jobs, i.e., the makespan. It is known that if the mechanism is non-preemptive, the price of anarchy is $\Omega(\log m)$. Both Azar, Jain, and Mirrokni (SODA 2008) and Caragiannis (SODA 2009) raised the question whether it is possible to design a coordination mechanism that has constant price of anarchy using preemption. We give a negative answer.

All deterministic coordination mechanisms, if they are symmetric and satisfy the property of independence of irrelevant alternatives, even with preemption, have the price of anarchy $\Omega(\frac{\log m}{\log \log m})$. Moreover, all randomized coordination mechanisms, if they are symmetric and unbiased, even with preemption, have similarly the price of anarchy $\Omega(\frac{\log m}{\log \log m})$.

Our lower bound complements the result of Caragiannis, whose BCOORD mechanism guarantees $O(\frac{\log m}{\log \log m})$ price of anarchy. Our lower bound construction is surprisingly simple. En route we prove a Ramsey-type graph theorem, which can be of independent interest.

On the positive side, we observe that our lower bound construction critically uses the fact that the inefficiency of a job on a machine can be unbounded. If, on the other hand, the inefficiency is not unbounded, we demonstrate that it is possible to break the $\Omega(\frac{\log m}{\log \log m})$ barrier on the price of anarchy by using known coordination mechanisms.

1 Introduction

The input is a set $\mathcal{I}$ of jobs and a set $\mathcal{M}$ of machines. Each job $i \in \mathcal{I}$ has a processing time p_{ij} on a machine $j \in \mathcal{M}$. Each job is controlled by a selfish player, who aims to minimize the completion time of his job while disregarding the welfare of other players. Each machine, based on the information of the incoming jobs and a certain scheduling policy, decides the finishing time of each incoming job. The scheduling policy of the machines is referred to as the *coordination mechanism* [4] in algorithmic game theory literature. The objective is to minimize the latest finishing time of any job. Such an objective is conventionally called the *makespan*.

[⋆] Part of this work is based on the first author's Master thesis in Max-Planck-Institut für Informatik.

The above scenario is very similar to the *unrelated machine scheduling problem* ($R||C_{max}$) that has been extensively studied in the literature, e.g.,[11]. However, unlike the traditional setting where a central authority decides which job is to be assigned to which machine, here we assume that each job is controlled by a selfish player. Our setting captures certain real world situations, such as the Internet, where there is no central authority and users are self-interested.

We assume that in the coordination mechanism, a machine is allowed to use only the information of the incoming jobs, when deciding how to schedule them, while the information of the other jobs and how the other machines schedule them are irrelevant. Using the terminology of [2], such coordination mechanisms are *local policies.*[3]

This assumption is a natural one: the machines may not be able to communicate among themselves efficiently to make a scheduling decision, especially in a very fluid environment such as the Internet.

Coordination mechanisms can be *non-preemptive* or *preemptive*. In the former, the jobs on a machine are processed sequentially, and each job, once started, has to be processed in an uninterrupted manner; in the latter, a machine can interrupt an ongoing job and resume it later, or it can intentionally introduce delays, during which the machine just lies idling.

Our focus will be on the *pure Nash equilibrium* (PNE), where no player can unilaterally change his strategy, i.e., the machine, to reduce the completion time of his job. It can be expected that given the selfishness of the players, the makespan in a PNE can be sub-optimal. The worst ratio of the makespan in a PNE against that in an optimal schedule is called the *price of anarchy* (PoA) [10].

Coordination mechanisms	PoA	PNE	Anonymous	Characteristics
ShortestFirst [9]	$\Theta(m)$	Yes	No	Strongly local, non-preemptive
LongestFirst [9]	Unbounded	No	No	Strongly local, non-preemptive
Makespan [9]	Unbounded	Yes	Yes	Strongly local, preemptive
RANDOM [9]	$\Theta(m)$	No	Yes	Strongly local, non-preemptive
EQUI [5]	$\Theta(m)$	Yes	Yes	Strongly local, preemptive
AJM-1 [2]	$\Theta(\log m)$	No	No	Local, non-preemptive
AJM-2 [2]	$O(\log^2 m)$	Yes	No	Local, preemptive
ACOORD [3]	$O(\log m)$	Yes	No	Local, preemptive
BCOORD [3]	$\Theta(\frac{\log m}{\log \log m})$	?	Yes	Local, preemptive
CCOORD [3]	$O(\log^2 m)$	Yes	Yes	Local, preemptive

Table 1. Summary of various coordination mechanisms.

[3] A more stringent assumption proposed by Azar et al. [2] is that of the *strongly local policies*. In this case, a machine makes the scheduling decision only by the processing times of the incoming jobs on it, while the processing times of these jobs on other machines are irrelevant. Azar et al. [2] have shown that strongly local policies have much higher lower bound in terms of the price of anarchy. In this work, we consider only local policies.

It is desirable to have coordination mechanisms that have small PoA. Table 1 gives a summary of the various mechanisms that have been proposed so far in the literature. The "PNE" column shows whether the existance of a pure Nash equilibrium is guaranteed or not. For non-preemptive coordination mechanisms, Azar, Jain, and Mirrokni [2] designed a mechanism that achieves $O(\log m)$ PoA. This turns out to be optimal, since later Fleischer and Svitkina [8] showed that all non-preemptive coordination mechanisms have $\Omega(\log m)$ PoA.

Since non-preemptive mechanisms have the $\Omega(\log m)$ PoA barrier, an obvious question to ask is whether preemption can beat this lower bound. Caragiannis [3] showed that using preemption, his BCOORD mechanism achieves $O(\frac{\log m}{\log \log m})$ PoA. Both Azar et al. and Caragiannis raised the question whether it is possible to achieve constant PoA by preemption. We answer in the negative. (See the next section for the formal definitions of "symmetric", "IIA", and "unbiased.")

Theorem 1. *All deterministic coordination mechanisms, if they are symmetric and satisfy independence of irrelevant alternatives (IIA) property, even with preemption, have the price of anarchy $\Omega(\frac{\log m}{\log \log m})$. Moreover, all randomized coordination mechanisms, if they are symmetric and unbiased, even with preemption, have similarly the price of anarchy $\Omega(\frac{\log m}{\log \log m})$. These lower bounds hold even for the special case of* restricted assignment ($B||C_{\max}$)*, where each job can go to at most 2 machines on which it has the processing time of 1.*

Therefore, the BCOORD mechanism of Caragiannis [3] is essentially the best possible. We prove this theorem in Section 2.

In our proof, we use the fact that a job can be assigned to only a subset of all machines, i.e., the restricted assignment model ($B||C_{\max}$). Let the *inefficiency* of a job i on a machine j be defined as $\frac{p_{ij}}{\min_{j' \in \mathcal{M}} p_{ij'}}$. The restricted assignment instances imply that the inefficiency of jobs on some machines is unbounded. This raises the issue whether it is possible to circumvent the $\Omega(\frac{\log m}{\log \log m})$ lower bound by assuming that inefficiency is bounded. We give a positive answer in Section 3. We show that the inefficiency-based mechanism [2] achieves $O(I)$ price of anarchy, where I is the largest possible inefficiency.

1.1 Our Assumptions and Technique

In proving our lower bounds, it is critical to first state our assumptions and definitions precisely. Recall that each job $i \in \mathcal{I}$ is associated with a *load characteristic* $\mathbf{p}_i = \langle p_{i1}, p_{i2}, \cdots, p_{i|\mathcal{M}|} \rangle$. If a job i cannot be processed at a machine j, let $p_{ij} = \infty$. Each job may or may not have an ID. When each job has a unique ID, we say these jobs are *non-anonymous.* When jobs do not have (unique) IDs, we say they are *anonymous.*

Our lower bound construction shares similar ideas to those of Azar et al. [2] and Fleischer and Svitkina [8]. For each machine, we give a set of jobs that are *indistinguishable* so as to confuse it.

Definition 1. *Let $j \in \mathcal{M}$ be a machine. Then two jobs i, i' are* indistinguishable *to j if the following holds.*

1. $p_{ij} = p_{i'j} = 1$,
2. *there exists two different machines* $j_i \neq j_{i'}$, $j \notin \{j_i, j_{i'}\}$ *and* $p_{ij_i} = p_{i'j_{i'}} = 1$,
3. $p_{ij*} = \infty$ *for* $j^* \in \mathcal{M}\backslash\{j, j_i\}$ *and* $p_{i'j*} = \infty$ *for* $j^* \in \mathcal{M}\backslash\{j, j_{i'}\}$.

A set of jobs are indistinguishable to machine j *if every two of them are indistinguishable to* j.

Definition 2. *Let* $\mathcal{C}$ *be a deterministic coordination mechanism.* $\mathcal{C}$ *is said to be* symmetric *if the following holds.*

Let $j, j' \in \mathcal{M}$ *be two different machines. Let* $\mathcal{I}_1$ *be a set of indistinguishable jobs to* j *and* $\mathcal{I}_2$ *a set of indistinguishable jobs to* j'. *Suppose that there exists a one-to-one correspondence* $\gamma : \mathcal{I}_1 \to \mathcal{I}_2$ *satisfying the following condition:*

> *For every job* $i \in \mathcal{I}_1$, *there exists a job* $\gamma(i) \in \mathcal{I}_2$ *so that* $p_{ij} = p_{\gamma(i)j'} = 1$. *Furthermore, there exists a permutation* $\sigma_i : \mathcal{M}\backslash\{j\} \to \mathcal{M}\backslash\{j'\}$ *so that* $p_{ij''} = p_{\gamma(i)\sigma_i(j'')}$ *for all* $j'' \in \mathcal{M}\backslash\{j\}$.

Then the set of the finishing times $t_{11} \leq t_{12} \leq \cdots \leq t_{1|\mathcal{I}_1|}$ *for* $\mathcal{I}_1$ *on machine* j *and the set of the finishing times* $t_{21} \leq t_{22} \leq \cdots \leq t_{2|\mathcal{I}_2|}$ *for* $\mathcal{I}_2$ *on machine* j' *are the same. i.e.,* $t_{1l'} = t_{2l'}$ *for* $1 \leq l' \leq |\mathcal{I}_1|$.

Intuitively speaking, a coordination mechanism is symmetric, if two machines, when they are presented with two sets of jobs that *look essentially the same*, then the finishing times for these two sets of jobs are the same on both machines. All coordination mechanisms in Table 1 are symmetric.

As a clarification, the above assumption states nothing regarding the *order* of the jobs to be finished on the machines. It is only about the *set* of their finishing times.

Definition 3. *Let* $\mathcal{C}$ *be a deterministic coordination mechanism.* $\mathcal{C}$ *is said to satisfy the* independence of irrelevant alternative *(IIA) property if the following holds.*

Let $j \in \mathcal{M}$ *be a machine and* i, i' *be two different jobs. Let* $\{i, i'\} \subseteq \mathcal{I}' \subset \mathcal{I}$. *If* j *is presented with the job set* $\mathcal{I}'$ *and it lets job* i *to be finished before* i', *then it also will let* i *to be finished before* i' *when it is presented with a job set* $\mathcal{I}' \cup \{k\}$ *for some job* $k \in \mathcal{I}\backslash\mathcal{I}'$.

Informally speaking, the IIA property states that if job i is "preferred" over i' by machine j, then this "preference" should not change because of the availability of some other jobs $k \notin \{i, i'\}$. The IIA property appears as an axiom in voting theory, bargaining theory, and logic [13].

The next lemma states that if a mechanism satisfies the IIA property, then each machine must have some sort of "preference list" over a set of indistinguishable jobs. The proof of the following lemma can be found in the appendix.

Lemma 1. *Let* $\mathcal{I}^*(j)$ *be a set of indistinguishable jobs to machine* j. *A deterministic coordination mechanism satisfies the IIA property iff, each machine*

$j \in \mathcal{M}$ has a strict linear order $\mathbb{L}_j$ over jobs in $\mathcal{I}^*(j)$, so that when j is presented with a subset $\mathcal{I}' \in \mathcal{I}^*(j)$ of these indistinguishable jobs, a job $i \in \mathcal{I}'$ has smaller completion than i' only when i precedes i' in the order $\mathbb{L}_j$.

Remark 1. We note that it is possible for a mechanism to satisfy the IIA property without "explicitly" having a strict linear order $\mathbb{L}_j$ over indistinguishable jobs: a machine can let all the indistinguishable jobs finish at the same time. This is indeed what several known deterministic mechanisms would have done, including MAKESPAN [9], BCOORD [3], and CCOORD [3]. In this case, the order $\mathbb{L}_j$ as stated in Lemma 1 can be just an arbitrary order over these indistinguishable jobs.

An IIA-satisfying deterministic mechanism could ask a machine to let all incoming indistinguishable jobs finish at the same time. But another possibility is that a machine j lets the incoming indistinguishable jobs finish at different times, when j does have an explicit linear order $\mathbb{L}_j$ over these indistinguishable jobs. An obvious candidate for $\mathbb{L}_j$ is an order over the job IDs when all jobs are non-anonymous. But even when jobs are anonymous, a machine still can use machine IDs of those machines on which these indistinguishable jobs can be processed to decide the order. To illustrate our point, assume that there are three machines, j_1, j_2, and j_3, and two jobs, i_1 and i_2. i_1 has the load characteristic $\langle 1, 1, \infty \rangle$ while i_2 has the load characteristic $\langle 1, \infty, 1 \rangle$. Even though these two jobs are indistinguishable to machine j_1, j_1 can deterministically choose to let i_1 finish first, if it prefers machine j_2 over j_3. By the above discussion, we make our assumption.

Definition 4. *Let $\mathcal{C}$ be a deterministic coordination mechanism satisfying the IIA property. Then the linear order $\mathbb{L}_j$ of each machine for a set of indistinguishable jobs as stated in Lemma 1 can take one of the following two forms.*

- ***Non-anonymous Case****: it is the preference list of machine j over the job IDs, or*
- ***Anonymous Case****: the preference list of machine j over the machine IDs. In particular, given two indistinguishable jobs i and i', i precedes i' in $\mathbb{L}_j$ if machine j prefers the machine j_i to $j_{i'}$, where j_i is the only other machine on which $p_{ij_i} = 1$ and $j_{i'}$ the only other machine on which $p_{i'j_{i'}} = 1$.*

Remark 2. Our assumptions stated in Definition 4 about the linear orders of the machines over the indistinguishable jobs are the same used as those by Azar et al. [2] and Fleischer and Svitkina [8] in their lower bound construction for non-preemptive coordination mechanisms. Suppose that machines do not have such preferences over the job IDs or the machine IDs, then a IIA-satisfying deterministic mechanism can only let all indistinguishable jobs finish at the same time (thus the linear order $\mathbb{L}_j$ of a machine j is an arbitrary order). We show that the same lower bound holds easily for this case in Section 2.3.

Sections 2.1 and 2.2 deal with the non-anonymous and anonymous cases respectively. The main technical challenge in our constructions is that the linear order $\mathbb{L}_j$ on the machines $j \in \mathcal{M}$ can differ from machine to machine. We need certain strategies to arrange the given set of jobs and machines so that in a PNE, the makespan is relatively high. Our lower bound construction for anonymous case is the most interesting part of this work. As a by-product, we derive a Ramsey-type graph theorem that has a similar flavor to the one obtained by Fleischer and Svitkina [8] when they proved their lower bound for non-preemptive mechanisms. In addition, our proof does not use Erdős-type probabilistic method; it is constructive and yields a polynomial time algorithm.

We now discuss our assumptions about randomized coordination mechanisms. It seems that there is not much work done concerning randomized mechanisms. The only one that we are aware of is the RANDOM mechanism of Immorlica et al. [9], which proceeds by ordering the incoming jobs of a machine uniformly at random and processing them non-preemptively. Cole et al. [6] used a randomized mechanism for the minimizing the weighted sum objective.

When randomized mechanisms are used, a PNE is an assignment in which no player can unilaterally change his machine to decrease the *expected* finishing time of his job.

Definition 5. *Let $\mathcal{C}$ be a randomized coordination mechanism.*

1. *$\mathcal{C}$ is* unbiased *if a machine $j \in \mathcal{M}$, when presented with a set of indistinguishable jobs, lets each of them have the same expected finishing time.*
2. *$\mathcal{C}$ is* symmetric *if two machines $j, j' \in \mathcal{M}$, when they are presented with the same number of indistinguishable jobs, let these two sets of indistinguishable jobs have the same set of expected finishing times.*

1.2 Related Work

The notion of price of anarchy, first introduced by Koutsoupias and Papadimitriou [10], plays a central role in the field of algorithmic game theory. A spate of papers have analyzed the PoA in various games. We refer the readers to [12] for an (incomplete) summary of them.

Machine scheduling as a field has received sustained attention from researchers for decades. When there is a central authority, in a classical paper, Lenstra, Shmoys, and Tardos [11] show that it is possible to achieve 2-approximation. Our lower bound offers an interesting contrast when jobs are controlled by selfish players.

The notion of coordination mechanisms was introduced by Christodoulou, Koutsoupias and Nanavati [4] in an effort to reduce the PoA. Table 1 summarizes of known coordination mechanisms for unrelated machine scheduling. For the more special cases of unrelated machine scheduling, such as restricted assignment and identical machines, better analysis of PoA for MAKESPAN mechanisms are given by [1, 7].

Finally, we note that even though the makespan is the most common objective in machine scheduling literature, there are other natural alternative objectives and it is worthwhile designing small PoA coordination mechanisms for them. Recently, Cole et al. [6] considered the objective of weighted sum of completion times of jobs.

2 Lower Bounds

All of our three lower bound constructions are based on a specific tree structure, which we will call the *equilibrium tree*. In such a tree, the root has k children, each of its k children has $k-1$ children, and each of these $k(k-1)$ grandchildren has $k-2$ children and so on. Generally, a vertex whose distance to the root is l has $k-l$ children. See Figure 1 as an illustration for the case of $k=3$. For convenience, we will use a bit unconventional terminology by referring to the root as the vertex in the k-th level, while its children are vertices in the $(k-1)$-st level and so on. Thus, a vertex in the l-th level has l children. We assume k to be some arbitrary large number.

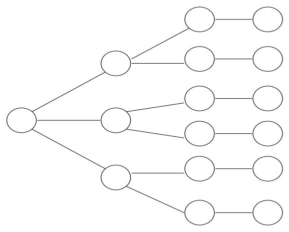

Fig. 1. The equilibrium tree with k=3

In all our constructions, a vertex in the equilibrium tree corresponds to a machine, while an edge (j,j') in the tree corresponds to a job i. Such a job has processing time $p_{ij}=p_{ij'}=1$, while $p_{ij''}=\infty$ for $j''\notin\{j,j'\}$. Suppose that j is in level t while j' is in level $t-1$, we say j is the parent machine (vertex) and j' is the child machine (vertex) of job $i=(j,j')$.

In our constructions, we will arrange the jobs and the machines corresponding to the equilibrium tree in such a way that in an optimal solution, all jobs will be assigned to their child machines, while in a PNE, all jobs are assigned to their parent machines. Clearly, in the optimal solution, the makespan is 1, while in the PNE, because there are k jobs on the root machines, the last job to be finished on it will have completion time at least k. Observe that the number of the vertices in the equilibrium tree is

$$\tilde{m}=1+\sum_{l=0}^{k}\prod_{s=0}^{l}(k-s)=1+k!(\sum_{s=0}^{k}\frac{1}{s!})<3(\frac{k}{e})^k\sqrt{2\pi k}<k^{2k}.$$

The function $f(x)=\frac{\ln x}{\ln\ln x}$ is strictly increasing when $x\geq e^e$. As we assume k to be large, both $\tilde{m}$ and k^{2k} are larger than e^e. So $f(\tilde{m})<f(k^{2k})$, implying

$$\frac{\ln\tilde{m}}{\ln\ln\tilde{m}}<\frac{2k\ln k}{\ln 2+\ln k+\ln\ln k}<\frac{2k\ln k}{\ln k}<2k.$$

Thus, if the number of the machines initially given is m and $m = \theta(\tilde{m})$, by the above inequality, we can conclude that the PoA in the constructed instance is at least $k = \Omega(\frac{\log m}{\log \log m})$.

2.1 Deterministic Mechanisms: Non-Anonymous Case

In this section, we assume that jobs are non-anonymous and a machine, when faced with a set of indistinguishable jobs, uses its preferences over the job IDs to decide the order of these indistinguishable jobs.

Let $m = \tilde{m}$, i.e., all m machines given will be part of the equilibrium tree. We assign the machines arbitrarily to the equilibrium tree and we will create $m-1$ jobs corresponding to their edges. Without loss of generality, let the job IDs to be from 1 to $m-1$. Recall that each machine may have a different preference order over these IDs. In the following, let X denote the set of job IDs that have been used in the algorithm.

We now apply the procedure in Figure 2 to construct the instance.

```
Let $X = \emptyset$.
For level $l$ from $k-1$ down to 0
   For each machine $j$ in level $l$
      Let $j'$ be the machine corresponding to $j$'s parent vertex.
      Choose $t$ to be the lowest ranking ID on $j$'s preference list that are not included
      in $X$.
      Create a job $i$ with ID $t$ and let $p_{ij} = p_{ij'} = 1$ and $p_{ij''} = \infty$ for $j'' \in \mathcal{M} \backslash \{j, j'\}$.
      $X := X \cup \{t\}$.
   End
End
```

Fig. 2. An algorithm to construct the equilibrium tree in the non-anonymous case.

Observe that by the algorithm, a machine prefers all the jobs that can be assigned to its vertices corresponding to its children in the equilibrium tree over the job that can be assigned to its parent in the equilibrium tree. This property will be used in the proof.

Theorem 2. *In the constructed instance, the PoA is* $\Omega(\frac{\log m}{\log \log m})$.

Proof. Clearly in the optimal assignment, each job should be assigned to the child machine. We now argue that if each job is assigned to its parent machine, we have a PNE. If this is the case, then the PoA is at least $k = \Omega(\frac{\log m}{\log \log m})$ and we have the proof.

So suppose not. Then there exists some job i between machine j at level l and machine j' at level $l-1$ and i has incentive to deviate from j to j'. Before the deviation, j has l incoming jobs that are indistinguishable; after the deviation, j' has similarly l incoming jobs that are indistinguishable. By

Definition 2, the set of complete times for these l incoming jobs in both cases are identical $t_1 \leq t_2 \leq \cdots \leq t_l$. By our construction, job i would have the completion time t_l after its deviation since its ID ranks lower than the IDs of all other $l-1$ jobs of machine j'. Before the deviation, job i has completion time $t_{l'}$ for some $1 \leq l' \leq l$. Since $t_{l'} \leq t_l$, we get a contradiction. □

2.2 Deterministic Mechanisms: Anonymous Case

In this section, we assume that jobs are anonymous and a machine, when faced with a set of indistinguishable jobs, uses its preferences over the machine IDs to decide the order of these indistinguishable jobs.

Assume that m machines are given, each with its own preference list over each other. (For convenience, the preference list over machine IDs can be interpreted as a preference order over other machines). We will choose a subset of machines ($\tilde{m}$ of them) and assign them to the equilibrium tree. Our goal is to make sure that each machine, if assigned to be a vertex in the equilibrium tree, ranks the machine corresponding to the parent vertex lower than all machines corresponding to its child vertices. We will discuss later how large m has to be (relative to $\tilde{m}$) so that such a construction is always possible. In the following, when the context is clear, we use the terms vertex and machine interchangeably.

Let n_s be the number of vertices in the s-th level in the equilibrium tree of totally k levels. Then

$$n_k = 1, \text{and } n_{l-1} = l n_l, \quad \forall 1 \leq l \leq k.$$

We will define another sequence n'_s for $0 \leq s \leq k$. Roughly speaking, this sequence denotes the numbers of vertices we will need in each level s in our construction.

We now describe our algorithm. It proceeds in $k-1$ iterations. In the beginning of each iteration l, we maintain n'_l equilibrium trees of l levels and n'_{l+1} equilibrium trees of $(l-1)$ levels. Let the roots of the former set be A and the roots of the latter set be B. We discard all vertices of the latter set of equilibrium trees, except for their roots, i.e., B. Let the vertices in B be $v_1, v_2, \cdots v_{n'_{l+1}}$ and we process them in this order. For v_1, choose the $l+1$ highest ranking vertices on v_1's preference list among all vertices in A. Make v_1 the parent of these roots. (So we have an equilibrium tree of $(l+1)$ levels rooted at v_1.) Remove these roots from A and we process v_2, v_3, and so on, in the same manner. At the end, all vertices in B are roots of equilibrium trees of $l+1$ levels, while the remaining vertices in A are the roots of the equilibrium trees of l levels. Note that if we make sure that

$$n'_l - (l+1)n'_{l+1} = n'_{l+2},$$

then in beginning of the next iteration, iteration $l+1$, we have the same condition as the the current iteration: n'_{l+1} equilibriums trees of $(l+1)$ levels and n'_{l+2} equilibrium trees of l levels.

We now formally define the sequence $\{n'_s\}_{s=0}^{k}$.

$$n'_k = n_k.$$
$$n'_{k-1} = kn'_k.$$
$$n'_{k-s} = (k-s+1)n'_{k-s+1} + n'_{k-s+2}, \quad \forall 2 \le s \le k.$$

We choose m to be $n'_0 + n'_1$. The full algorithm is presented in Figure 3.

```
Out of the $m$ given vertices, choose $n'_1$ arbitrary vertices and denote them as $B$ and
the rest as $A$.
For each vertex $v$ in $B$
  Choose the highest ranking vertex $v' \in A$.
  Make $v$ the parent of $v'$.
  $A = A \backslash \{v'\}$.
End            // The prepartion is done.
For level $l$ from 1 up to $k-1$
  Let the roots of the equilibrium trees of $(l-1)$ levels be $B$; throw away all other
  vertices in these trees.
  Let the roots of the equilibrium trees of $l$ levels be $A$.
  For each vertex $v$ in $B$
    Choose $(l+1)$ highest ranking vertices $v_1, v_2, \cdots, v_{l+1}$ among all vertices in
    $A$ on $v$'s preference list.
    Make $v$ the parents of $v_1, v_2, \cdots, v_{l+1}$.
    $A = A \backslash \{v_i\}_{i=1}^{l+1}$.
  End
End
```

Fig. 3. An algorithm to construct the equilibrium tree in the anonymous case.

Lemma 2. *The final outcome of the above algorithm is an equilibrium tree of k levels; moreover, in such a tree, every non-leaf/non-root vertex ranks all of its child vertices higher than its parent vertex.*

Proof. We prove by establishing the following claim.

Claim 1 *In the beginning of iteration l, $1 \le l \le k-1$, there are n'_l equilibrium trees of l levels and n'_{l+1} equilibrium trees of $l-1$ levels. Moreover, each root of the former ranks its child vertices higher than any of the roots of the latter.*

Proof. We prove by induction. The base case $l = 1$ holds trivially based on what the first for loop of the algorithm and the fact that $n'_0 - n'_1 = n'_2$. By induction hypothesis, in the beginning of the $(l-1)$-st iteration, there are n'_{l-1} equilibrium trees of $l-1$ levels, and n'_l equilibrium trees of $l-2$ levels. At the end of $(l-1)$-st iteration, the latter set is thrown away except their roots. ln'_l of the former will be merged with these roots into n'_l equilibrium trees of l levels. So there are only $n'_{l-1} - ln'_l = n'_{l+1}$ equilibrium trees of $(l-1)$ levels left. This completes the first part of the induction step. The second part of the induction step follows trivially from the way we choose to merge the equilibrium trees. □

By the first part of the above claim, in the beginning of the last iteration, we have n'_{k-1} equilibrium trees of $k-1$ levels and n'_k equilibrium trees of $k-2$ levels. By the algorithm, at the end of the last iteration, we have $n'_k = 1$ equilibrium tree of k levels and $n'_{k-1} - kn'_k = 0$ equilibrium trees of $k-1$ levels. So we are left with exactly an equilibrium tree of k levels. For the second part of the lemma, choose any vertex v at level l. Observe that such a vertex must be a root in the beginning of iteration l and its parent u must be one of the roots of those equilibrium trees of $l-1$ levels. By the second part of the above claim, we conclude that v prefers its child vertices to u. The lemma follows. □

We now bound m by establishing the following lemma.

Lemma 3. *$n'_l < n_l + n'_{l+1}$, for each $0 \le l \le k-1$.*

Proof. We prove by induction. Let the base case be $k-1$. Then $n'_{k-1} = kn'_k = kn_k = n_{k-1} < n_{k-1} + n'_k$. For the induction step,

$$n'_l = (l+1)n'_{l+1} + n'_{l+2} < (l+1)(n_{l+1} + n'_{l+2}) + n'_{l+2} = n_l + (l+2)n'_{l+2} \le n_l + n'_{l+1},$$

where the first inequality follows from induction hypothesis. So the lemma follows. □

Lemma 4. *$\tilde{m} \le m \le 2\tilde{m}$.*

Proof. The first inequality holds because by Lemma 3, after throwing away some vertices from the given m vertices, the algorithm ends up with an equilibrium tree of k levels, whose number of vertices is exactly $\tilde{m}$.

For the second inequality, by the definition n'_k and the previous lemma, we know that

$$\begin{aligned} n'_k &\le n_k \\ n'_l &\le n_l + n'_{l+1} \qquad \text{for all } 0 \le l \le k-1 \end{aligned}$$

Summing up the above inequalities, we have $n'_0 \le \sum_{l=0}^{k} n_l = \tilde{m}$. The lemma holds because

$$m = n'_0 + n'_1 < 2n'_0 \le 2\tilde{m}.$$

□

Theorem 3. *In the constructed instance, the PoA is $\Omega(\frac{\log m}{\log\log m})$.*

Proof. Clearly in the optimal assignment, each job should be assigned to the child machine. We now argue that if each job is assigned to its parent machine, we have a PNE. If this is the case, then the PoA is at least $k = \Omega(\frac{\log \tilde{m}}{\log\log \tilde{m}}) =$

$\Omega(\frac{\log m}{\log\log m})$, where the second equality follows from Lemma 4, and we would have the proof.

So suppose not. Then there exists some job i between machine j at level l and machine j' at level $l-1$ and i has incentive to deviate from j to j'. Before the deviation, j has l incoming jobs that are indistinguishable; after the deviation, j' has similarly l incoming jobs that are indistinguishable. By Definition 2, the set of complete times for these l incoming jobs in both cases are identical $t_1 \leq t_2 \leq \cdots \leq t_l$. By Lemma 3, machine j' prefers all child vertices over j, therefore, it also prefers all its other incoming jobs over i. This implies that job i would have the completion time t_l after its deviation. Before the deviation, job i has completion time $t_{l'}$ for some $1 \leq l' \leq l$. Since $t_{l'} \leq t_l$, we arrive at a contradiction. □

The following corollary follows from Lemmas 3 and 4.

Corollary 1. *Let T be a tree of the following property: the root has k children, and the vertex whose distance to the root is l has $k-l$ children itself.*

Let $G=(V,E)$ be a graph, where each vertex in V has a strictly-ordered preference over other vertices. Suppose that $|V| \geq 2|T|$. Then we can always find a subset of vertices $V' \subset V$, $|V'| = |T|$, and assign these vertices to T so that a vertex $u \in V'$ prefers the vertices in V' corresponding to its children in T to the vertex in V' corresponding to its parent in T.

2.3 Deterministic Mechanisms: When Machines Do Not Use Job or Machine IDs

Suppose that machines do not use job or machine IDs to break ties when it is faced with a set of indistinguishable jobs, then the only possibility for scheduling these jobs is to let them finish at the same time. In this case, we can use the same construction to get the lower bound very easily.

Let $m = \tilde{m}$ and assign all machines to the equilibrium tree of k levels arbitrarily. For each edge (j,j') in the tree, create a job with $p_{ij} = p_{ij'} = 1$ and $p_{ij''} = \infty$ for $j'' \notin \{j,j'\}$. To see that all jobs assigned to their parent machines result in a PNE, observe that if a job deviate to its child machine, the number of the jobs on that machine would be the same as the number of jobs on its parent machine before its deviation, therefore the deviating job would have the same finishing time on both machines, due to Definition 2. We can thus conclude that the PoA in this case is also $\Omega(\frac{\log m}{\log\log m})$.

2.4 Randomized Mechanisms

Consider the same instance used in the preceding section. We argue that if all jobs are assigned to their parent machines, the outcome would be a PNE. Observe that if a job i deviates to its child machine in level $l-1$, then this child machine is faced with a set of l indistinguishable jobs. On the other hand, before the deviation of i, its parent machine is also faced with a set of l indistinguishable jobs. Since we assume that machines are unbiased and symmetric, by

Definition 5, the expected completion time of job i would be identical in both cases. We can thus conclude that the PoA in this case is also $\Omega(\frac{\log m}{\log\log m})$.

3 Upper Bound on Price of Anarchy When Inefficiency Is Bounded

In this section, we demonstrate that the $\Omega(\frac{\log m}{\log\log m})$ lower bound on PoA can be circumvented if the inefficiency of the jobs on the machines is bounded by I.

We analyze the upper bound of PoA of the inefficiency-based mechanism proposed by Azar et al. [2]. Let $p_i = \min_{j\in\mathcal{M}} p_{ij}$, the minimum processing time of a job on all machines. In this mechanism, each machine $j \in M$ non-preemptively processes the incoming jobs based on nondecreasing order of their inefficiency on it: that is, given two jobs i and i', if $\frac{p_{ij}}{p_i} < \frac{p_{i'j}}{p_{i'}}$, then job i should be processed before j (ties are broken by job IDs). This rather intuitive mechanism turns out to be optimal for non-preemptive mechanism. As shown by Azar et al. [2], its PoA is $O(\log m)$, matching the lower bound of non-preemptive mechanism.

Theorem 4. *The inefficiency-based mechanism has PoA at most $I + 2\log I + 2$.*

Proof. Given a PNE, let j_1 be the most loaded machine, whose load is x, and j_2 be the least loaded machine, whose load is y. Furthermore, let i^* be the last job finished on machine j_1.

Observe that

$$x - y \le p_{i^*j_2} \le Ip_{i^*} \le I\mathbf{OPT}.$$

The first inequality holds because if not, then job i^* has incentive to migrate to machine j_2, a contradiction to the assumption that we are given a PNE; the second inequality holds because of the assumption that inefficiency is bounded by I; the third inequality holds because the makespan of the optimal assignment can not be less than the minimum weight of job i^*. Now

$$x \le I\mathbf{OPT} + y.$$

In the following, we prove that $y \le (2\log I + 2)\mathbf{OPT}$. Thus dividing the above inequality by $\mathbf{OPT}$ proves the theorem. □

Claim 2 *$y \le (2\log I + 2)\boldsymbol{OPT}$.*

Proof. The proof of the claim uses some ideas from [2].

Divide the interval $[0, y]$ into $k = \lfloor \frac{y}{2\mathbf{OPT}} \rfloor$ contiguous levels, each of which has length of $2\mathbf{OPT}$. The last part of the interval $[0, y]$ whose length is $y - k\mathbf{OPT} < 2\mathbf{OPT}$ does not belong to any level. If $k = 0$, then $y < 2\mathbf{OPT}$ and the proof follows easily. So in the following, we assume that $k \ge 1$.

Let M_{kj} be all jobs (and parts of jobs) that are processed on machine j that are processed after time $2k\mathbf{OPT}$. Let $M_k = \cup_{j\in\mathcal{M}} M_{jk}$. Let R_{kj} be the sum of the minimum weight of jobs in M_{kj}. Precisely,

$$R_{kj} = \sum_{i \in M_{kj}} p_i * \frac{\text{amount of time after } 2k\mathbf{OPT} \text{ machine } j \text{ processes job } i}{p_{ij}}$$

(Observe that in fact there is at most one job i in M_{kj} whose contribution to R_{kj} is less than its minimum processing time p_i.)

Let $R_k = \sum_{j \in \mathcal{M}} R_{kj}$. Observe that

$$\frac{R_0}{m} \leq \mathbf{OPT}, \tag{1}$$

since R_0 is the sum of the minimum processing times of all jobs.

Now let A_k be the "average inefficiency" of all jobs that are processed in the interval $[2(k-1)\mathbf{OPT}, 2k\mathbf{OPT}]$, that is,

$$A_k = \frac{2m\mathbf{OPT}}{R_{k-1} - R_k},$$

where the numerator is the total amount of work done by all the machines in the interval $[2(k-1)\mathbf{OPT}, 2k\mathbf{OPT}]$ and the denominator is sum of the minimum weight of all jobs that are (partially) processed during this interval. By the definition of R_k and the assumption that all jobs have inefficiency of at most I, we have

$$A_k \leq I, \tag{2}$$

Next we use a lemma proved by Azar et al. [2].

Lemma 5. [2, Lemma 4.2] *$R_k \leq (1/2)R_{k-1}$ for all $k \geq 1$.*

By this lemma, we have

$$A_k = \frac{2m\mathbf{OPT}}{R_{k-1} - R_k} > \frac{2m\mathbf{OPT}}{R_{k-1}} \geq \frac{2m\mathbf{OPT}}{R_0(1/2)^{k-1}} = 2^k\mathbf{OPT}\frac{m}{R_0}. \tag{3}$$

Combining Inequalities (1),(2), and (3), we have

$$2^k\mathbf{OPT} \leq \frac{R_0}{m}A_k \leq \frac{R_0}{m}I \leq I\mathbf{OPT},$$

implying that $k \leq \log I$. We can thus conclude that $y < (2k+2)\mathbf{OPT} \leq (2\log I + 2)\mathbf{OPT}$, and the proof is complete. □

Acknowledgments

We thank Kurt Mehlhorn, Zoya Svitkina, and Reto Spöhel for their helpful discussions.

References

1. Baruch Awerbuch, Yossi Azar, Yossi Richter, and Dekel Tsur. Tradeoffs in worst-case equilibria. *Theor. Comput. Sci.*, 361(2-3):200–209, 2006.
2. Yossi Azar, Kamal Jain, and Vahab S. Mirrokni. (almost) optimal coordination mechanisms for unrelated machine scheduling. In *SODA*, pages 323–332, 2008.
3. Ioannis Caragiannis. Efficient coordination mechanisms for unrelated machine scheduling. In *SODA*, pages 815–824, 2009.
4. George Christodoulou, Elias Koutsoupias, and Akash Nanavati. Coordination mechanisms. In *ICALP*, pages 345–357, 2004.
5. Johanne Cohen, Christoph Dürr, and Nguyen Kim Thang. Non-clairvoyant scheduling games. *Theory Comput. Syst.*, 49(1):3–23, 2011.
6. Richard Cole, José R. Correa, Vasilis Gkatzelis, Vahab S. Mirrokni, and Neil Olver. Inner product spaces for minsum coordination mechanisms. In *STOC*, pages 539–548, 2011.
7. Artur Czumaj and Berthold Vöcking. Tight bounds for worst-case equilibria. *ACM Transactions on Algorithms*, 3(1), 2007.
8. Lisa Fleischer and Zoya Svitkina. Preference-constrained oriented matching. In *Proceedings of the Seventh Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 66–73, 2010.
9. Nicole Immorlica, Li (Erran) Li, Vahab S. Mirrokni, and Andreas S. Schulz. Coordination mechanisms for selfish scheduling. *Theor. Comput. Sci.*, 410(17):1589–1598, 2009.
10. Elias Koutsoupias and Christos H. Papadimitriou. Worst-case equilibria. *Computer Science Review*, 3(2):65–69, 2009.
11. Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46:259–271, 1990.
12. N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.
13. Wikipedia. http://en.wikipedia.org/wiki/Independence of irrelevant alternatives.

A Proof of Lemma 1

The ($\leftarrow$) direction is obvious. For the ($\rightarrow$) direction, we create the strict linear order $\mathbb{L}_j$ for machine j as follows.

For every two indistinguishable jobs i and i' in $\mathcal{I}^*(j)$, if a machine j, given any subset $\mathcal{I}' \subseteq \mathcal{I}^*(j)$ and $\mathcal{I}' \supseteq \{i, i'\}$, lets i to have smaller completion time than i'. Let i precedes i' in $\tilde{\mathbb{L}}_i$. Due to the IIA property, the precedence order in $\tilde{\mathbb{L}}_j$ must be transitive. So $\tilde{\mathbb{L}}_j$ is a linear order with possibly ties. Let $\mathbb{L}_j$ be derived from $\tilde{\mathbb{L}}_j$ by breaking ties in $\tilde{\mathbb{L}}_j$ arbitrarily.

To see that $\mathbb{L}_j$ satisfies the property stated in the lemma, note that if job i has smaller completion time than i' when machine j is faced with $\mathcal{I}' \subseteq \mathcal{I}^*(j)$ and $\mathcal{I}' \supseteq \{i, i'\}$, then i precedes i' in $\tilde{\mathbb{L}}_j$, hence also in $\mathbb{L}_j$. $\square$