Coordinating Selfish Players in Scheduling Games

Fidaa Abed

Saarbrücken, Germany

Dissertation
zur Erlangung des Grades des
Doktors der Ingenieurswissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

April 2015

To my parents

Abstract

We investigate coordination mechanisms that schedule n jobs on m unrelated machines. The objective is to minimize the makespan. It was raised as an open question whether it is possible to design a coordination mechanism that has constant price of anarchy using preemption. We give a negative answer. Next we introduce multi-job players that control a set of jobs, with the aim of minimizing the sum of the completion times of theirs jobs. In this setting, previous mechanisms designed for players with single jobs are inadequate, e.g., having large price of anarchy, or not guaranteeing pure Nash equilibria. To meet this challenge, we design three mechanisms that induce pure Nash equilibria while guaranteeing relatively small price of anarchy.

Then we consider multi-job players where each player's objective is to minimize the weighted sum of completion time of her jobs, while the social cost is the sum of players' costs. We first prove that if machines order jobs according to Smith-rule, then the coordination ratio is at most 4, moreover this is best possible among non-preemptive policies. Then we design a preemptive policy, *externality* that has coordination ratio 2.618, and complement this result by proving that this ratio is best possible even if we allow for randomization or full information. An interesting consequence of our results is that an $\varepsilon-$local optima of $R| \,| \sum w_i C_i$ for the jump neighborhood can be found in polynomial time and is within a factor of 2.618 of the optimal solution.

Kurzfassung

Wir betrachten Koordinationsmechanismen um n Jobs auf m Maschinen mit individuellen Bearbeitungszeiten zu verteilen. Ziel dabei ist es den Makespan zu minimieren. Es war eine offene Frage, ob es möglich ist einen preämptiven Koordinationsmechanismus zu entwickeln, der einen konstanten Price of Anarchy hat. Wir beantworten diese Frage im negativen Sinne. Als nächstes führen wir Multi-Job-Spieler ein, die eine Menge von Jobs kontrollieren können, mit dem Ziel die Summe der Fertigstellungszeiten ihrer Jobs zu minimieren. In diesem Szenario sind bekannte Mechanismen, die für Ein-Job-Spieler entworfen worden sind, nicht gut genug, und haben beispielsweise einen hohen Price of Anarchy oder können kein reines Nash Gleichgewicht garantieren. Wir entwickeln drei Mechanismen die jeweils ein reines Nash Gleichgewicht besitzen, und einen relativ kleinen Price of Anarchy haben.

Zusätzlich betrachten wir Multi-Job-Spieler, mit dem Ziel jeweils die gewichtete Summe der Fertigstellungszeiten ihrer Jobs zu minimieren, während die Gesamtkosten die Summe der Kosten der Spieler sind. Wir zeigen zuerst, dass das Koordinationsverhältnis höchstens 4 ist, wenn die Maschinen die Jobs nach der Smith-Regel sortieren, was bei nicht-preämptiven Verfahren optimal ist. Danach entwickeln wir ein preämptives Verfahren, *Externality*, welches ein Koordinationsverhältnis von 2.618 hat, und ergänzen dieses Ergebniss indem wir beweisen, dass dieses Verhältnis optimal ist, auch für den Fall, dass wir Randomisierung oder volle Information erlauben. Eine interessante Folge unserer Ergebnisse ist, dass ein ε-lokales Optimum von $R||\sum w_iC_i$ für die Jump-Neighborhood in Polynomialzeit gefunden werden kann, und innerhalb eines Faktors von 2.618 von der optimalen Lösung ist.

Acknowledgements

All thanks to my God who gives me everything good in this life.

I thank my parents and my wife for their support.

I also thank my son and my daughter who give me the motivation in my way.

I would like to thank my supervisor Kurt Mehlhorn for providing me the opportunity to work in his group and for allowing me to pursue my own line of research. His guidance was of immense help.

I also thank my hardworking advisor Chien-Chung Huang. He was a perfect advisor for me. I am really pround to be his first student.

I should not forget to thank my friends Mahmoud Fouz, Mohammed Shaheen, and Eyad Alkassar. Their help was a main reason for me to join the master's and PhD program at MPI-INF.

Contents

Chapter 1

Introduction

Machine scheduling originates in the optimization of manufacturing systems and their formal mathematical treatment dates back to at least the pioneering work of Smith (52). In general, scheduling problems can be described as follows. Consider a set $\mathcal{J}$ of n jobs that have to be processed on a set $\mathcal{M}$ of m parallel machines. If processed on machine j, job i requires a certain processing time p_{ij} to be completed. Job i also may have a weight w_i and, in addition, it may have other characteristics such as release dates, time windows, delays when switching a task from one machine to another, or precedence constraints. The goal is to find an assignment of jobs to machines, and an ordering within each machine so that a certain objective functions is minimized. Denoting, for any such assignment and ordering, the *completion time* C_i of job i the time at which job i completes, we may write the two most widely studied objectives as $C_{\max} = \max_{i \in \mathcal{J}} C_i$ (the makespan) and $\sum_{i \in \mathcal{J}} w_i C_i$ (the sum of weighted completion times). In terms of the machine environment the most basic model is that of *identical* machines, where the processing times of jobs are the same on all machines. In the *related* machines environment each machine has a speed, and the processing time of a job on a machine is inversely proportional to the speed of that machine. Finally in *unrelated* machine scheduling the processing times are arbitrary, thus capturing all the above models as special cases. The unrelated machines environment with the makespan objective (denoted by $R| \,| C_{\max}$), and sum of weighted completion times objective (denoted by $R| \,| \sum w_i C_i$) are the focus of this thesis.

Since the early work of Smith, a lot of work has been put in designing *centralized* algorithms providing reasonably close to optimal solutions with limited computational effort for these NP-hard problems (10; 22; 26; 32; 33; 34; 37; 46;

47; 48; 49; 50; 51). The underlying assumption is that all information is gathered by a single entity which can enforce a particular schedule. However, as distributed environments emerge, understanding scheduling problems where jobs are managed by different selfish agents (players), who are interested in their own completion time, becomes a central question.

1.1 Coordination Mechanisms

In recent times there has been quite some effort to understand these scheduling games in the special case in which agents control a single job in the system, which we call *single-job games*. In this context, there is a vast amount of work studying existence, uniqueness, the *price of anarchy* (36), and other characteristics of equilibrium when, given some processing rules, each agent seeks to minimize her own completion time. In the scheduling game each job is a fully informed player wanting to minimize its individual completion time, and its set of strategies correspond to the set of machines. Job i's completion time on a machine depends on the strategies chosen by other players, and on the *policy* (or processing rule) of the chosen machine. A *coordination mechanism* is then a set of *local policies*, one per machine, specifying how the jobs assigned to that machine are scheduled. In a *local policy* the schedule on a machine depends on the full vector $(p_{i1}, p_{i2}, \dots, p_{im})$ and weights w_i of jobs assigned to that machine. In contrast, in a *strongly local policy* the schedule on machine j must be a function only of the processing times p_{ij} and weights w_i of the jobs assigned to j. In evaluating the efficiency of these policies, one needs a benchmark to compare this social cost against. The definition of the price of anarchy of the induced game considers a social optimum with respect to the costs specified by the chosen machine policies. However, to measure the quality of a coordination mechanism we consider the worst case ratio of the social cost at an equilibrium to the optimal social cost that could be achieved by the centralized optimization approach. We refer to this as the *coordination ratio* or the *price of anarchy* of a mechanism.

The following two conditions on coordination mechanisms are (implicitly) assumed by all previous works (and the current one).

1. **Physical Feasibility**. Suppose that a set of jobs $\mathcal{J}' \subseteq \mathcal{J}$ are assigned to machine j. At any point of time t, if a subset of jobs $\mathcal{J}'' \subseteq \mathcal{J}'$ are finished by machine j, then $\sum_{i \in \mathcal{J}''} p_{ij} \leq t$.

2. **Locality of Scheduling Decision**. A machine decides its schedule based only on the information of the incoming jobs, while, where the other jobs go to, and how the other machines schedule them is irrelevant.

The first condition is self-evident; the second condition is motivated by the fact that in a fluid environment, such as the Internet, a machine may not be able to coordinate with other machines in a timely manner.

1.2 Machine Policies

We say a policy is prompt if it does not introduce deliberate idle time. In other words, if jobs $i_1, \ldots, i_k$ are assigned to machine j, then by time $\sum_{\ell=1}^{k} p_{i_\ell,j}$ all jobs have been completed and released. Besides distinguishing between local and strongly local policies we distinguish between *non-preemptive*, *preemptive*, and *randomized* policies. In non-preemptive policies jobs are processed in some fixed deterministic order that may depend arbitrarily on the set of jobs assigned to the machine (processing time, weight, and ID), and once a job is completed it is released. On the other hand, preemptive policies may suspend a job before it completes in order to execute another job and the suspended job is resumed later. Interestingly, such policies can be considered as non-preemptive policies, but where jobs may be held back after completion (17; 18). Finally, randomized policies have the additional power that they can schedule jobs at random according to some distribution depending on the assigned jobs' characteristics. Another usual distinction is between policies that are *anonymous* and *non-anonymous*. In the former jobs with the same characteristics (except for IDs) must be treated equally and thus assigned the same completion time. In the latter, jobs may be distinguished using their IDs.

For instance consider the widely used policy known as Smith-rule (**sr**), which sorts jobs in nondecreasing order of their processing time to weight ratios. Formally **sr** processes jobs in nondecreasing order of $\rho_{ij} = p_{ij}/w_i$, and breaks ties using the job's IDs. This policy is strongly local, non-preemptive, and non-anonymous.

1.3 Related Literature

The notion of price of anarchy, first introduced by Koutsoupias and Papadimitriou (36), 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 (41) for an (incomplete) summary of them.

The study of coordination mechanisms for single-job scheduling games, taking the makespan as social cost, was initiated by Christodoulou et al. (14). However the implied bounds on the price of anarchy are constant only for simple environments such as when machines are identical. Indeed, Azar et al. (6), and Fleischer and Svitkina (27) show that, even for a restricted uniform machines environment "almost" no non-preemptive deterministic machine policies satisfying the so-called "independence of irrelevant alternatives" property can achieve a constant price of anarchy. The existence of a randomized machine policy with such a desirable property is unknown. It is worth mentioning that there is a vast amount of related work considering the makespan social cost (11; 12; 21; 23; 35; 38).

The situation changes quite dramatically for the sum of weighted completion times objective. In this case Correa and Queyranne show that, for restricted related machines, smith rule induces a game with price of anarchy at most 4 (20), improving results implied by Farzad et al. (24) and Caragiannis et al (12) obtained in different contexts. Cole et al, extend this result to unrelated machines, and also design an improved preemptive policy, proportional sharing, achieving an approximation bound of 2.618 and an even better randomized policy (17; 18). Further recent works include extensions and improvements by Bhattacharya et al (8), Cohen et al (16) and by Rahn and Schäfer (43), Hoeksma and Uetz (31).

Finally, performance guarantee results for the $\sum w_i C_i$ objective using natural local search heuristics are scarse, despite the vast amount of computational work (13; 42). We are only aware of the results of Brueggemann et al. (9) who proved that for identical machines local optima for the jump neighborhood are within a factor of 3/2 of the optimal schedule.

1.4 Contribution and Organization

The thesis is organized as follows. Each chapter consists of one model together with its detailed notation and obtained results.

Chapter 2: This chapter deals with the model where each player controls one job (single-job game). The player's goal is to minimize the completion time of her job while the global objective is to minimize 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.

Indication of source: The content of Chapter 2 has been previously published in ESA 2012 (2).

Chapter 3: This chapter introduces the model where each player controls a set of jobs (multi-job game). The player's goal is to minimize the sum of completion times of her jobs while the global objective is to minimize the makespan. In this setting, previous mechanisms designed for players with single jobs are inadequate, e.g., having large price of anarchy, or not guaranteeing pure Nash equilibria. To meet this challenge, we design three mechanisms that are adapted/generalized from Caragiannis' ACOORD. All our mechanisms induce pure Nash equilibria while guaranteeing relatively small price of anarchy.

Indication of source: The content of Chapter 3 has been previously published in TCS journal 2015 (3).

Chapter 4: This chapter deals with the model where each player controls a set of jobs(multi-job game). The player's goal is to minimize the weighted sum of completion times of her jobs while the global objective is to minimize the weighted sum of completion times of all jobs. We work with a weaker equilibrium concept that includes that of Nash.

We first prove that if machines order jobs according to their processing time to weight ratio, a.k.a. Smith-rule, then the coordination ratio is at most 4, moreover this is best possible among non-preemptive policies. Then we establish our main result. We design a preemptive policy, *externality*, that extends Smith-rule by adding extra delays on the jobs accounting for the negative externality they impose on other players. For this policy we prove that the coordination ratio is $1 + \phi \approx 2.618$, and complement this result by proving that this ratio is best possible even if we allow for randomization or full information. Finally, we establish that this externality policy induces a potential game and that an ε-equilibrium can be found in polynomial time. An interesting consequence of our results is that an $\varepsilon-$local optima of $R||\sum w_iC_i$ for the jump (a.k.a. move) neighborhood can be found in polynomial time and are within a factor of 2.618 of the optimal solution.

Indication of source: The content of Chapter 4 has been previously published in ESA 2014 (1).

Chapter 2

Makespan Minimization in Single-Job Games

2.1 Introduction

The input is a set $\mathcal{J}$ of jobs and a set $\mathcal{M}$ of machines. Each job $i \in \mathcal{J}$ 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 her 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* (14) 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*.

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.,(37). 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 (6), such coordination mechanisms are *local policies*.[1]

[1]A more stringent assumption proposed by Azar et al. (6) is that of the *strongly local*

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.

Recall that 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.

In this chapter, our focus will be on the *pure Nash equilibrium* (PNE), where no player can unilaterally change her strategy, i.e., the machine, to reduce the completion time of her 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) (36).

Coordination mechanisms	PoA	PNE	Anonymous	Characteristics
ShortestFirst (35)	$\Theta(m)$	Yes	No	Strongly local, non-preemptive
LongestFirst (35)	Unbounded	No	No	Strongly local, non-preemptive
MAKESPAN (35)	Unbounded	Yes	Yes	Strongly local, preemptive
RANDOM (35)	$\Theta(m)$	No	Yes	Strongly local, non-preemptive
EQUI (15)	$\Theta(m)$	Yes	Yes	Strongly local, preemptive
AJM-1 (6)	$\Theta(\log m)$	No	No	Local, non-preemptive
AJM-2 (6)	$O(\log^2 m)$	Yes	No	Local, preemptive
ACOORD (11)	$O(\log m)$	Yes	No	Local, preemptive
BCOORD (11)	$\Theta(\frac{\log m}{\log\log m})$	?	Yes	Local, preemptive
CCOORD (11)	$O(\log^2 m)$	Yes	Yes	Local, preemptive

Table 2.1: Summary of various coordination mechanisms.

It is desirable to have coordination mechanisms that have small PoA. Table 3.1 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 (6) designed a mechanism that achieves $O(\log m)$ PoA. This turns out to be optimal, since later Fleischer and Svitkina (27) showed that all non-preemptive coordination mechanisms have $\Omega(\log m)$ PoA.

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. (6) have shown that strongly local policies have much higher lower bound in terms of the price of anarchy. In this chapter, we consider only local policies.

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 (11) showed that using preemption, her 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 2.1.1. *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 (11) 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 (6) achieves $O(I)$ price of anarchy, where I is the largest possible inefficiency.

2.2 Assumptions and Technique

In proving our lower bounds, it is critical to first state our assumptions and definitions precisely. Recall that each job $i \in \mathcal{J}$ 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. (6) and Fleischer and Svitkina (27). For each machine, we give a set of jobs that are *indistinguishable* so as to confuse it.

Definition 2.2.1. *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 2.2.2. *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 $\mathcal{J}_1$ be a set of indistinguishable jobs to j and $\mathcal{J}_2$ a set of indistinguishable jobs to j'. Suppose that there exists a one-to-one correspondence $\gamma : \mathcal{J}_1 \to \mathcal{J}_2$ satisfying the following condition:

> *For every job $i \in \mathcal{J}_1$, there exists a job $\gamma(i) \in \mathcal{J}_2$ 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 $t_{11} \leq t_{12} \leq \cdots \leq t_{1|\mathcal{J}_1|}$ for $\mathcal{J}_1$ on machine j and the set of the finishing times $t_{21} \leq t_{22} \leq \cdots \leq t_{2|\mathcal{J}_2|}$ for $\mathcal{J}_2$ on machine j' are the same. i.e., $t_{1l'} = t_{2l'}$ for $1 \leq l' \leq |\mathcal{J}_1|$.

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 3.1 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 2.2.3. *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 $\{i, i'\} \subseteq \mathcal{J}' \subset \mathcal{J}$. If j is presented with the job set $\mathcal{J}'$ 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 $\mathcal{J}' \cup \{k\}$ for some job $k \in \mathcal{J}\backslash\mathcal{J}'$.

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 (54).

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.

Lemma 2.2.4. *Let $\mathcal{J}^*(j)$ 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 $\mathcal{J}^*(j)$, so that when j is presented with a subset $\mathcal{J}' \in \mathcal{J}^*(j)$ of these indistinguishable jobs, a job $i \in \mathcal{J}'$ has smaller completion than i' only when i precedes i' in the order $\mathbb{L}_j$.*

Proof. 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 $\mathcal{J}^*(j)$, if a machine j, given any subset $\mathcal{J}' \subseteq \mathcal{J}^*(j)$ and $\mathcal{J}' \supseteq \{i, i'\}$, 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 $\mathcal{J}' \subseteq \mathcal{J}^*(j)$ and $\mathcal{J}' \supseteq \{i, i'\}$, then i precedes i' in $\tilde{\mathbb{L}}_j$, hence also in $\mathbb{L}_j$.

$\square$

Remark 2.2.5. *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 *(35),* BCOORD *(11), and* CCOORD *(11). In this case, the order $\mathbb{L}_j$ as stated in Lemma 2.2.4 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 2.2.6. *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 2.2.4 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 2.2.7. *Our assumptions stated in Definition 2.2.6 about the linear orders of the machines over the indistinguishable jobs are the same used as those by Azar et al. (6) and Fleischer and Svitkina (27) 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 (27) 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. (35), which proceeds by ordering the incoming jobs of a machine uniformly at random and processing them non-preemptively. Cole et al. (17) 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 her machine to decrease the *expected* finishing time of her job.

Definition 2.2.8. *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.*

2.3 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 4.1 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.

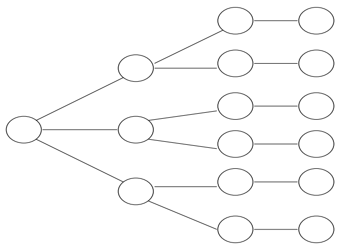

Figure 2.1: The equilibrium tree with k=3

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 $k = \Omega(\frac{\log m}{\log \log m})$.

2.3.1 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 2.2 to construct the instance.

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

```
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
```

Figure 2.2: An algorithm to construct the equilibrium tree in the non-anonymous case.

the job that can be assigned to its parent in the equilibrium tree. This property will be used in the proof.

Theorem 2.3.1. *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 2.2.2, 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.

□

2.3.2 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 $\{n'_s\}_{s=0}^{k}$.

$$\begin{aligned} n'_k &= n_k. \\ n'_{k-1} &= k n'_k. \\ n'_{k-s} &= (k-s+1)n'_{k-s+1} + n'_{k-s+2}, \quad \forall 2 \leq s \leq k. \end{aligned}$$

We choose m to be $n'_0 + n'_1$. The full algorithm is presented in Figure 2.3.

```
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
```

Figure 2.3: An algorithm to construct the equilibrium tree in the anonymous case.

Lemma 2.3.2. *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 2.3.3. *$n'_l < n_l + n'_{l+1}$, for each $0 \leq l \leq k-1$.*

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,

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

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

Lemma 2.3.4. *$\tilde{m} \leq m \leq 2\tilde{m}$.*

Proof. The first inequality holds because by Lemma 2.3.3, 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

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

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

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

□

Theorem 2.3.5. *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 2.3.4, 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 2.2.2, 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 2.3.3, 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 2.3.3 and 2.3.4.

Corollary 2.3.6. *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.

2.3.3 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 2.2.2. We can thus conclude that the PoA in this case is also $\Omega(\frac{\log m}{\log \log m})$.

2.3.4 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 2.2.8, 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 $\Omega(\frac{\log m}{\log \log m})$.

2.4 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. (6). 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. (6), its PoA is $O(\log m)$, matching the lower bound of non-preemptive mechanism.

Theorem 2.4.1. *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

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

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

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

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

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

Proof. The proof of the claim uses some ideas from (6).

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 $k \geq 1$.

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

$$\frac{R_0}{m} \leq \mathbf{OPT}, \tag{2.1}$$

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

$$A_k \leq I, \tag{2.2}$$

Next we use a lemma proved by Azar et al. (6).

Lemma 2.4.2. (6, Lemma 4.2) $R_k \leq (1/2)R_{k-1}$ *for all* $k \geq 1$.

By this lemma, we have

$$A_k = \frac{2m\mathbf{OPT}}{R_{k-1} - R_k} > \frac{2m\mathbf{OPT}}{R_{k-1}} \geq \frac{2m\mathbf{OPT}}{R_0(1/2)^{k-1}} = 2^k\mathbf{OPT}\frac{m}{R_0}. \tag{2.3}$$

Combining Inequalities (2.1),(2.2), and (2.3), 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.

$\square$

$\square$

Chapter 3

Makespan Minimization in Multi-Job Games

3.1 Introduction

All previous works assume that a player controls a single job. A natural and more realistic extension is to assume a player can control multiple jobs and we refer to such players as multi-job players. A question that arises in our model is: what would be the local objective of a multi-job player? This is a non-issue when a player controls a single job. However, when she has multiple jobs, several objectives are possible. For instance, it could be her makespan (the latest finishing time of her jobs), or it could be the sum of completion times of her jobs.

In this chapter, we assume that each player aims to minimize the sum of the completion times of her jobs. This assumption is motivated by the observation that a player would care about the *collective welfare* of her jobs. If moving a job from one machine to another machine decreases the finishing time of that job, the controlling player would have incentive to do so—even if the latest finishing time of her jobs is not really decreased.

To evaluate the overall system performance, there can be two natural candidates: makespan (the latest finishing time of a job), or the weighted completion times of the jobs (jobs are given weights and the cost is computed as the weighted sum of their completion times.) In the next chapter of this thesis, we use the weighted completion times of the jobs to measure the system performance. In this chapter, we instead consider the makespan.

In general, in terms of PoA, it is harder to design mechanisms when the global objective is the makespan than when it is the weighted sum of completion times

Mechanisms	PoA		PNE	
	$C = 1$	$C > 1$	$C = 1$	$C > 1$
ShortestFirst (35)	$\Theta(m)$	$\Omega(m)$	Yes	No$\star$
LongestFirst (35)	Unbounded	Unbounded	No	No
Makespan (35)	Unbounded	Unbounded	Yes	No$\star$
RANDOM (35)	$\Theta(m)$	$\Omega(m)$	No	No
EQUI (15)	$\Theta(m)$	$\Omega(m)$	Yes	Yes
AJM-1 (6)	$\Theta(\log m)$	$\Omega(\log m)$	No	No
AJM-2 (6)	$\Theta(\log^2 m)$	$\Omega(\log^2 m)$	Yes	No$\star$
BALANCE (16)	$\Theta(\log m)$	$\Omega(\log m)$	Yes	No$\star$
ACOORD (11)	$O(p \cdot m^{1/p})$	$\Omega(C^{(1-\epsilon)(p+1)}m/p^2)\star$	Yes	Yes
BCOORD (11)	$O(p \cdot m^{1/p}/\log p)$	$\Omega(C^{(1-\epsilon)(p+1)}m/p^2)\star$	No	No$\star$
CCOORD (11)	$O(p^2 \cdot m^{1/p})$	$\Omega(C^{(1-\epsilon)(p+1)}m/p^2)$ when $p = 1\star$	Yes	No$\star$

Table 3.1: Summary of the properties of known mechanisms in our model. $m = |\mathcal{J}|$ is the number of machines, and C is the largest number of jobs controlled by a player. The results marked by $\star$ are proved in this chapter. For the last three mechanisms, $p \geq 1$, and ϵ is some small constant where $\epsilon > 0$. If $p = \Theta(\log m)$ and $C = 1$, then the PoAs for ACOORD, BCOORD, and CCOORD are $\Theta(\log m)$, $\Theta(\frac{\log m}{\log \log m})$, and $O(\log^2 m)$ respectively.

of all jobs. In the original model where each player controls a single job, with weighted sum global objective, Cole et al. (17) show several mechanisms achieving constant PoA; on the other hand, when the global objective is the makespan, it is known (2; 6; 27) that constant PoA is impossible. As multi-job model is a generalization of the single-job model, we also cannot hope to achieve constant PoA.

We observe that previous mechanisms designed for players with single jobs are inadequate in multi-job model. In some cases (ACOORD/ BCOORD/ CCOORD), the PoA becomes significantly worse; in some cases, they no longer guarantee PNEs (ShortestFirst/AJM-2/CCOORD/BALANCE). See Table 3.1 for a summary of the properties of the known mechanisms in multi-job model. Our challenge here is to design coordination mechanisms that simultaneously guarantee the existence of PNEs *and* still maintain small PoA.

3.2 Contribution

We propose three mechanisms, A_1-COORD, A_2-COORD, and A_3-COORD, which are presented in Sections 4-6. These mechanisms make use of preemption and

Mechanisms	PoA	PNE	Note
A_1-COORD	$O(C^{q+1}m^{\frac{1}{q+1}})$ for any $q > 0$	Yes	Local
A_2-COORD	$O(C^{\frac{2q}{q+1}}m^{\frac{1}{q+1}})$ for any $0 < q \leq 1$, $1/q \in \mathbb{Z}$	Yes	Local
A_3-COORD	$O(\min\{W\sqrt{m}, \min_{\gamma\in\mathbb{Z}_{\geq 1}}\{m^{\frac{\gamma+1}{2\gamma}} + W^\gamma\}\})$, $O(\log m + \log W)$ when $C = 1$	Yes	Strongly local

Table 3.2: Summary of our mechanisms. $m = |\mathcal{J}|$ is the number of machines, and C is the largest number of jobs controlled by a player. In A_3-COORD, $W = \frac{\max_{i\in\mathcal{J}}\min_{j\in\mathcal{J}} p_{ij}}{\min_{i\in\mathcal{J}}\min_{j\in\mathcal{J}} p_{ij}}$. In A_1-COORD and A_2-COORD, PNEs can be computed in polynomial time.

assume that players and jobs are not anonymous, namely, each player and each job has a unique ID. When a job is assigned to a machine, the machine can make schedule decisions based on this job's ID and the ID of its owner.

Our mechanisms are adapted/generalized from Caragiannis' ACOORD (hence the naming). See Table 3.2 for a summary of their properties. All of them induce PNEs. Under A_1-COORD and A_2-COORD, such PNEs can be computed in polynomial time; furthermore, each player can compute her own optimal strategy in polynomial time.

In terms of PoA, our three mechanisms perform differently depending on the situation. Let $m = |\mathcal{J}|$ be the number of machines and C be the largest number of jobs controlled by a player. A_1-COORD achieves the PoA of $O(C^{q+1}m^{\frac{1}{q+1}})$, for any chosen $q > 0$. A_1-COORD is better suited for the situation when C is some bounded constant (in this case we can get a PoA of $O(m^\epsilon)$). When C is relatively large, A_2-COORD is a better mechanism, with the PoA of $O(C^{\frac{2q}{q+1}}m^{\frac{1}{q+1}})$, for any chosen q, $0 < q \leq 1$ so that $1/q$ is an integer. For example, if m is bounded by a constant and C is very large, then we can get a PoA of $O(C^\epsilon)$.

Our third mechanism, A_3-COORD, has the PoA independent of C and, in some cases, is superior to the previous two. Let $W = \frac{\max_{i\in\mathcal{J}}\min_{j\in\mathcal{J}} p_{ij}}{\min_{i\in\mathcal{J}}\min_{j\in\mathcal{J}} p_{ij}}$, i.e., the largest ratio of sizes of two jobs when they are both assigned to the most efficient machines. Then A_3-COORD guarantees the PoA of $O(\min\{W\sqrt{m}, \min_{\gamma\in\mathbb{Z}_{\geq 1}}\{m^{\frac{\gamma+1}{2\gamma}} + W^\gamma\}\})$. Unlike the previous two mechanisms that are local, A_3-COORD is strongly local, thus more "frugal" in terms of the information it needs. Additionally, when $C = 1$, (i.e., the single-job model), A_3-COORD achieves the PoA of $O(\log m + \log W)$. Previously, Cohen, Dürr, and Thang (15) raised the question whether it is possible to design a strongly local mechanism that achieves the PoA of $O(polylog(m))$. Here we give a partial positive answer—as long as $W = O(m^{polylog(m)})$.

How our mechanisms guarantee PNEs is similar to the original ACOORD[1], using a simple idea: the finishing times of the jobs of the k-th player is dependent only on the strategies of the first $k-1$ players and the k-th player himself. This idea also ensures the game converges to PNEs in polynomial steps. The main technical challenge of this work is in the analysis of PoA. To prove that our mechanisms have the claimed PoAs, we introduce several non-trivial extensions of Caragiannis' ideas in the analysis of his ACOORD.

3.3 Related Work

Our three mechanisms are adapted from ACOORD mechanism (11). This mechanism uses a global ordering of the jobs according to their distinct IDs. The finishing time of a job is the total load of the jobs preceding it and itself, modified by a certain *inefficiency* parameter. The game induced by this mechanism is a potential game which guarantees the existence of a PNE. Moreover, the convergence to a PNE is fast. Our three mechanisms are generalized from ACOORD by fine tuning the inefficiency parameter.

The notion of multi-job players is similar to the notion of "oligopolistic players". Though so far not directly considered in the machine scheduling context, the notion of "oligopolistic players" has in fact been studied in different settings. For instance, in a version of selfish routing (44), an atomic player controls a splittable flow. Such a player can be regarded as an oligopolistic player. See (7; 19) and the references therein for an overview of such games. Another example of an oligopolistic player is a coalition of players. In (4; 25), a partition equilibrium, where the agents are partitioned into coalitions, and only deviations by the prescribed coalitions are considered, is proved to exist in resource selection games. In (28; 30), the authors assume that once a set of players form a coalition, they care only about their collective welfare while disregarding their own outcomes (thus there is no backstabbing or double-crossing). A coalition, under such assumptions, is equivalent to an oligopolistic player.

Table 3.1 summarizes the performance of various known mechanisms in multi-job setting. As mentioned before, the difficulty is to guarantee both the existence

[1]We note that when $c = 1$, A_1-COORD and A_2-COORD reduce to ACOORD.

of PNEs and a small PoA. Only ACOORD and EQUI guarantee PNEs in multi-job model. To ensure that ACOORD has a PNE, we just need to index jobs in such a way that all jobs belonging to the same player have consecutive indices. EQUI was originally designed to guarantee a *strong* Nash equilibrium, when players control single jobs. Interestingly, in multi-job model, it can be shown that it still induces a potential game (thus guaranteeing PNEs). We leave it as an open question regarding its real PoA when $C > 1$.

3.4 Preliminary

We first introduce some necessary notations to facilitate our discussion. Throughout the chapter, we use N to denote an assignment and O the optimal assignment. N_j(resp. O_j) is the set of jobs assigned to machine j in N(resp. O). $L(N_j) = \sum_{i \in N_j} p_{ij}$ is the total load of jobs assigned to j in assignment N. For each job i, let $p_{i,\min}$ be its smallest size, $p_{i,\min} = \min_{j \in \mathcal{J}} p_{ij}$, and ϕ_{ij} its *inefficiency* on machine j, defined as $\frac{p_{ij}}{p_{i,\min}}$. Note that only local mechanisms can make use of the inefficiencies of the jobs in scheduling, while strongly local mechanisms cannot. We assume that the set $\mathcal{P}$ of players are indexed consecutively, from 1, 2, $\cdots$, up to $|\mathcal{P}|$. Given job $i \in \mathcal{I}$, $\pi(i)$ denotes the player controlling it.

Proposition 3.4.1. (11) *For an assignment N and any $p \geq 1$,*
$\max_{j \in \mathcal{J}} L(N_j) \leq (\sum_{j \in \mathcal{J}} L(N_j)^p)^{\frac{1}{p}} \leq m^{\frac{1}{p}} \max_{j \in \mathcal{J}} L(N_j)$.

Proposition 3.4.2. *Let $a_i, b_i \geq 0$, $p \geq 1$, and let $f(x)$ be a convex function.*

- *Minkowski's inequality:* $(\sum_{i=1}^{s}(a_i + b_i)^p)^{1/p} \leq (\sum_{i=1}^{s} a_i^p)^{1/p} + (\sum_{i=1}^{s} b_i^p)^{1/p}$.
- *Jensen's inequality:* $\sum_{i=1}^{s} f(a_i) \geq s f(\frac{\sum_{i=1}^{s} a_i}{s})$.

The next proposition is an easy consequence of Minkowski's inequality.

Proposition 3.4.3. *Let $p \geq 1$ be some integer. Then $a^{1/p} + b^{1/p} \geq (a+b)^{1/p}$.*

Proof. By re-writing $a^{1/p}$ as $((a^{1/p})^p)^{1/p}$ and $b^{1/p}$ as $((b^{1/p})^p)^{1/p}$, we can apply Minkowski's inequality

$$((a^{1/p})^p)^{1/p} + ((b^{1/p})^p)^{1/p} \geq ((a^{1/p} + b^{1/p})^p)^{1/p}$$
$$= [a + b + \sum_{t=1}^{p-1} \left(\begin{array}{c} p \\ t \end{array} \right) (a^{1/p})^t (b^{1/p})^{p-t}]^{1/p} \geq (a+b)^{1/p}.$$

□

Proposition 3.4.4. (11) *Suppose that $p \geq 1$, $A \geq 0$, and $B_i \geq 0$ for $i = 1, \cdots, s$. Then $\sum_{i=1}^{s}((A + B_i)^p - A^p) \leq (A + \sum_{i=1}^{s} B_i)^p - A^p$.*

The following proposition is slightly modified from Caragiannis (11). Specifically, we replace the constrain on the exponent $p \geq 1$ with $p \geq 0$ so as to design a larger set of mechanisms. The proof is entirely the same as in (11).

Proposition 3.4.5. (11) *For any $z_0 \geq 0$, $\alpha \geq 0$, and $p \geq 0$, the following holds.*

$$(p + 1)\alpha z_0^p \leq (z_0 + \alpha)^{p+1} - z_0^{p+1} \leq (p + 1)\alpha(z_0 + \alpha)^p.$$

Proof. If $\alpha = 0$, the lemma holds trivially. If $\alpha > 0$, the lemma holds by observing that the function z^{p+1} is convex for any $p \geq 0$. Therefore, the slope of the line crossing (z_0, z_0^{p+1}) and $(z_0 + \alpha, (z_0 + \alpha)^{p+1})$ is between its derivatives at points z_0 and $z_0 + \alpha$. □

Proposition 3.4.6. *Let $A \geq 0$, $B_i \geq 0$ for $1 \leq i \leq s$, and $p \geq 0$. Then*

$$(A + \sum_{i=1}^{s} B_i)^{1+p} - A^{1+p} \leq (1 + p)s^p \sum_{i=1}^{s} B_i(A + B_i)^p.$$

Proof. Observe that the function $g(x) = x(A + x)^p$ is convex when $x \geq 0$, since $g''(x) = p(A + x)^{p-2}(2A + xp + x) \geq 0$. Therefore,

$$\sum_{i=1}^{s} B_i(A + B_i)^p = \sum_{i=1}^{s} g(B_i) \geq sg(\frac{\sum_{i=1}^{s} B_i}{s}) = \sum_{i=1}^{s} B_i(A + \frac{\sum_{t=1}^{s} B_t}{s})^p,$$

where the inequality follows from the convexity of $g(x)$ and Jensen's inequality.

Using the above inequality, we have

$$
\begin{aligned}
& (1+p)s^p \sum_{i=1}^{s} B_i(A+B_i)^p \\
\geq\ & (1+p)s^p \sum_{i=1}^{s} B_i(A+\frac{\sum_{t=1}^{s} B_t}{s})^p \\
=\ & (1+p) \sum_{i=1}^{s} B_i(As+\sum_{t=1}^{s} B_t)^p \\
\geq\ & (1+p) \sum_{i=1}^{s} B_i(A+\sum_{t=1}^{s} B_t)^p \\
=\ & (1+p)(\sum_{i=1}^{s} B_i)(A+\sum_{i=1}^{s} B_i)^p \\
\geq\ & (A+\sum_{i=1}^{s} B_i)^{1+p} - A^{1+p},
\end{aligned}
$$

where the last inequality follows from Proposition 3.4.5 by setting $\sum_{i=1}^{s} B_i = \alpha$, and $z_0 = A$. The proof follows.

$\square$

3.5 A_1-COORD

In this and the next section, let N_j^k denote the set of jobs assigned to j belonging to the first k players, for any $k \in \mathcal{P}$. Observe that $N_j^{|\mathcal{P}|} = N_j$ and $N_j^0 = \emptyset$. Finally, let $L(N_j^k) = \sum_{i:\pi(i)\leq k, i\in N_j} p_{ij}$, the total load of jobs belonging to the first k players on machine j in N.

> A_1-COORD: Let N be the assignment. Suppose that job $i \in N_j$. Then the completion of job i is set as $P(i, N_j) = C(\phi_{ij})^{\frac{1}{q}}(L(N_j^{\pi(i)-1})+p_{ij})$, for some $q > 0$.

As in the original ACOORD, the term $(\phi_{ij})^{\frac{1}{q}}$ is used to encourage a player to assign her job to a more efficient machine. The term $L(N_j^{\pi(i)-1})$ is the total load of jobs belonging to the first $\pi(i)-1$ players on machine j. So a job's completion time is unaffected by jobs belonging to players with indices larger than $\pi(i)$. This property will be used when we argue that A_1-COORD has a PNE.

The important idea behind our mechanism is that the jobs belonging to the same player, even if they are assigned to the same machine j, *would have their completion times independent of each other*. This follows from the simple observation that the sum $L(N_j^{\pi(i)-1}) + p_{ij}$ does not include other jobs belonging to the player $\pi(i)$. This property is a key part in our analysis of PoA; also it allows each player to compute her own optimal strategy in polynomial time (see Theorem 3.5.4). Finally, the multiplicative factor C is introduced to make sure that A_1-COORD produces feasible schedules.

Lemma 3.5.1. *The schedule decided by A_1-COORD is feasible.*

Proof. Recall that to prove a schedule is feasible, we need to show that at time f, if a set $\mathcal{I}' \subseteq N_j$ of jobs are finished, then $\sum_{i \in \mathcal{I}'} p_{ij} \leq f$. It is easy to see that we only need to consider those times f where some job $i \in N_j$ are finished.

Now suppose that in assignment N player k puts jobs i_1, i_2, $\cdots$, $i_{x \leq C}$ on machine j, where the jobs are indexed by their non-decreasing completion times. We argue that the completion time $P(i_y, N_j)$ of job i_y, $y \leq x$, is at least as large as the total load of jobs finishing no later than i_y. In the case that multiple jobs among i_1, i_2, $\cdots$, $i_{x \leq C}$ finish at the same time as i_y, w.l.o.g., we can assume that i_y has the largest index. Consider two possibilities.

1. Suppose that none of the jobs belonging to players in $\mathcal{P} \backslash \{1, 2, \cdots, k\}$ finishes earlier than i_y. Since jobs $i_1, \cdots, i_{y-1}$ finish no later than i_y, $(\phi_{i_t j})^{1/q}(L(N_j^{k-1}) + p_{i_t j}) \leq (\phi_{i_y j})^{1/q}(L(N_j^{k-1}) + p_{i_y j})$, for $1 \leq t \leq y-1$. As a result,

$$P(i_y, N_j) = C(\phi_{i_y j})^{1/q}(L(N_j^{k-1}) + p_{i_y j}) \geq$$
$$\sum_{t=1}^{y} (\phi_{i_t j})^{1/q}(L(N_j^{k-1}) + p_{i_t j}) \geq L(N_j^{k-1}) + \sum_{t=1}^{y} p_{i_t j},$$

   which is no less than the total load of jobs finishing as early as i_y.

2. Suppose that some players in $\mathcal{P} \backslash \{1, 2, \cdots, k\}$ have jobs on machine j that finish earlier than i_y. Let player $k+s$ be such player with the largest index. Suppose further that player $k+s$ has jobs i_1^*, i_2^*, $\cdots$, i_z^* finish as early as i_y on machine j. W.L.O.G., assume that i_1^* is the job with the largest size among the jobs i_1^*, i_2^*, $\cdots$, i_z^*. Then

$$P(i_y, N_j) \geq P(i_1^*, N_j) = C(\phi_{i_1^* j})^{1/q}(L(N_j^{k+s-1}) + p_{i_1^* j}) \geq$$
$$(\phi_{i_1^* j})^{1/q}(L(N_j^{k+s-1}) + \sum_{t=1}^{z} p_{i_t^* j}) \geq L(N_j^{k+s-1}) + \sum_{t=1}^{z} p_{i_t^* j},$$

which is no less than the total load of jobs finishing as early as i_y. The proof follows.

□

Lemma 3.5.2. *Suppose that N is a PNE under A_1-COORD. Then* $\max_{j\in\mathcal{J},i\in N_j} P(i,N_j) \le C[(\sum_{j\in\mathcal{J}} L(N_j)^{q+1})^{\frac{1}{q+1}} + \max_{j\in\mathcal{J}} L(O_j)]$.

Proof. Let i^* be the job with the largest completion time in assignment N. Suppose that it is assigned to j_1 in assignment N and has an inefficiency 1 on machine j_2 (j_2 could be the same as j_1). Either $j_2 \neq j_1$, then the player $\pi(i^*)$ controlling i^* has no incentive to move i^* to j_2; or $j_2 = j_1$. In both cases, we have

$$P(i^*, N_{j_1}) \le C(L(N_{j_2}^{\pi(i^*)-1}) + p_{i^*j_2}) \le$$
$$C(L(N_{j_2}) + p_{i^*j_2}) \le C(\sum_{j\in\mathcal{J}} L(N_j)^{q+1})^{\frac{1}{q+1}} + C\max_{j\in\mathcal{J}} L(O_j),$$

where the last inequality follows from Proposition 3.4.1 and the fact that in the optimal, one machine would have load at least $p_{i^*j_2}$. The proof follows.

□

Lemma 3.5.3. *Suppose that N is a PNE under A_1-COORD. Then*

$$(\sum_{j\in\mathcal{J}} L(N_j)^{q+1})^{\frac{1}{q+1}} \le 4(q+1)C^q m^{\frac{1}{q+1}} \max_{j\in\mathcal{J}} L(O_j).$$

Proof. Suppose that job i is assigned to machine j_1 in assignment N and machine j_2 in the optimal assignment O. As N is a PNE, $C(\phi_{ij_1})^{1/q}(L(N_{j1}^{\pi(i)-1}) + p_{ij_1}) \le C(\phi_{ij_2})^{1/q}(L(N_{j2}^{\pi(i)-1}) + p_{ij_2})$. Canceling C, raising both sides to the power of q, and multiplying them by $p_{i,\min}$, we have

$$p_{ij_1}(L(N_{j_1}^{\pi(i)-1}) + p_{ij_1})^q \le p_{ij_2}(L(N_{j_2}^{\pi(i)-1}) + p_{ij_2})^q \tag{3.1}$$

Define $x_{ij} = 1(0)$ if job i is (not) assigned to machine j in assignment N; similarly define $y_{ij} = 1(0)$ if job i is (not) assigned to machine j in the optimal O. Then the above inequality can be re-written as

$$\begin{aligned}\sum_{j\in\mathcal{J}} x_{ij}p_{ij}(L(N_j^{\pi(i)-1}) + x_{ij}p_{ij})^q &\le \sum_{j\in\mathcal{J}} y_{ij}p_{ij}(L(N_j^{\pi(i)-1}) + y_{ij}p_{ij})^q \\ &\le \sum_{j\in\mathcal{J}} y_{ij}p_{ij}(L(N_j) + y_{ij}p_{ij})^q.\end{aligned}$$

Summing the above inequality over all jobs $i \in \mathcal{I}$, we have

$$\begin{aligned}
&(q+1)\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} x_{ij}p_{ij}(L(N_j^{\pi(i)-1}) + x_{ij}p_{ij})^q \\
\leq\ & \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}(q+1)y_{ij}p_{ij}(L(N_j) + y_{ij}p_{ij})^q \\
\leq\ & \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}[L(N_j) + 2y_{ij}p_{ij})^{q+1} - (L(N_j) + y_{ij}p_{ij}]^{q+1} \\
\leq\ & \sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{I}}[L(N_j) + 2y_{ij}p_{ij})^{q+1} - (L(N_j))]^{q+1} \\
\leq\ & \sum_{j\in\mathcal{J}}[L(N_j) + 2\sum_{i\in\mathcal{I}} y_{ij}p_{ij})^{q+1} - (L(N_j))]^{q+1} \\
\leq\ & [(\sum_{j\in\mathcal{J}} L(N_j)^{q+1})^{\frac{1}{q+1}} + 2(\sum_{j\in\mathcal{J}} L(O_j)^{q+1})^{\frac{1}{q+1}}]^{q+1} - \sum_{j\in\mathcal{J}} L(N_j)^{q+1}, \qquad (3.2)
\end{aligned}$$

where the second inequality follows from Proposition 3.4.5 by setting $\alpha = y_{ij}p_{ij}$ and $z_0 = L(N_j) + y_{ij}p_{ij}$, and $p = q$; the fourth inequality from Proposition 3.4.4 by setting $A = L(N_j)$, $B_i = 2y_{ij}p_{ij}$ and $p = 1+q$; and the fifth inequality from Minkowski's inequality.

We next bound $\sum_{j\in\mathcal{J}} L(N_j)^{q+1}$ by writing it as a telescopic sum:

$$\begin{aligned}
\sum_{j\in\mathcal{J}} L(N_j)^{q+1} &= \sum_{j\in\mathcal{J}}\sum_{k=1}^{|\mathcal{P}|} L(N_j^k)^{q+1} - L(N_j^{k-1})^{q+1} \\
&= \sum_{j\in\mathcal{J}}\sum_{k=1}^{|\mathcal{P}|}[L(N_j^{k-1}) + \sum_{i:\pi(i)=k, i\in N_j} p_{ij}]^{q+1} - L(N_j^{k-1})^{q+1} \\
&\leq \sum_{j\in\mathcal{J}}\sum_{k=1}^{|\mathcal{P}|}\sum_{i:\pi(i)=k, i\in N_j}(1+q)|\{i|\pi(i)=k, i\in N_j\}|^q p_{ij}(L(N_j^{k-1}) + p_{ij})^q \\
&\leq \sum_{j\in\mathcal{J}}\sum_{k=1}^{|\mathcal{P}|}\sum_{i:\pi(i)=k, i\in N_j}(1+q)C^q p_{ij}(L(N_j^{k-1}) + p_{ij})^q \\
&= (1+q)C^q\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} x_{ij}p_{ij}(L(N_j^{\pi(i)-1}) + x_{ij}p_{ij})^q \qquad (3.3) \\
&\leq C^q\{[(\sum_{j\in\mathcal{J}} L(N_j)^{q+1})^{\frac{1}{q+1}} + 2(\sum_{j\in\mathcal{J}} L(O_j)^{q+1})^{\frac{1}{q+1}}]^{q+1} - \sum_{j\in\mathcal{J}} L(N_j)^{q+1}\} \qquad (3.4)
\end{aligned}$$

where the first inequality follows from Proposition 3.4.6 by setting $A = L(N_j^{k-1})$, $B_i = p_{ij}$ and $p = q$; the second inequality from the fact that a player has at most

C jobs on machine j in assignment N; the third equality from a double counting argument; and the last inequality from (3.2).

Rearranging terms in Inequality (3.4), we have

$$(\sum_{j \in \mathcal{J}} L(N_j)^{q+1})^{\frac{1}{q+1}} \leq \frac{2(\sum_{j\in\mathcal{J}} L(O_j)^{q+1})^{\frac{1}{q+1}}}{(\frac{1}{C^q}+1)^{\frac{1}{q+1}}-1} = \frac{2(\sum_{j\in\mathcal{J}} L(O_j)^{q+1})^{\frac{1}{q+1}}}{((\frac{1}{C^q}+1)^{C^q})^{\frac{1}{C^q(q+1)}}-1}$$
$$\leq \frac{2m^{\frac{1}{q+1}} \max_{j\in\mathcal{J}} L(O_j)}{((\frac{1}{C^q}+1)^{C^q})^{\frac{1}{C^q(q+1)}}-1},$$

where the second inequality follows from Proposition 3.4.1.

Observe that as $C \geq 1$, $C^q \geq 1$. Then by calculus, $\sqrt{e} < 2 \leq (\frac{1}{C^q}+1)^{C^q} \leq e$. Thus, the term $(\sum_{j\in\mathcal{J}} L(N_j)^{q+1})^{\frac{1}{q+1}}$ can be further upper-bounded as follows,

$$(\sum_{j \in \mathcal{J}} L(N_j)^{q+1})^{\frac{1}{q+1}} \leq \frac{2m^{\frac{1}{q+1}} \max_{j\in\mathcal{J}} L(O_j)}{((\frac{1}{C^q}+1)^{C^q})^{\frac{1}{C^q(q+1)}}-1} < \frac{2m^{\frac{1}{q+1}} \max_{j\in\mathcal{J}} L(O_j)}{e^{\frac{1}{2C^q(q+1)}}-1}$$
$$\leq 4(q+1)C^q m^{\frac{1}{q+1}} \max_{j\in\mathcal{J}} L(O_j),$$

where the last inequality holds because of the well-known inequality that $e^z - 1 \geq z$. Hence the proof.

□

Theorem 3.5.4. *A_1-COORD guarantees a PNE. Such a PNE can be computed in polynomial time and each player can compute her optimal strategy in polynomial time. Moreover, for any fixed $q > 0$, it guarantees that PoA of $O(C^{q+1}m^{\frac{1}{q+1}})$.*

Proof. For the first part, we can construct a PNE as follows. Let all players 1, 2, $\cdots$, $|\mathcal{P}|$, in this order, choose their optimal strategies one at a time. For any player k, his strategy under A_1-COORD is only dependent on the strategies of previous players. No matter how the later players choose their strategies, player k has no reason to deviate. Therefore, the outcome is a PNE.

To see that each player can compute her optimal strategy in polynomial time and the aforementioned PNE can be constructed in polynomial time, observe that under A_1-COORD, each of her jobs has completion time independent of her other jobs. Therefore, she can simply assign each of her jobs to the machine that causes the least finishing time of that job. The outcome of such assignment is clearly her optimal strategy and can be computed in polynomial time.

The last part of the theorem follows from Lemmas 3.5.2 and 3.5.3.

□

3.6 A_2-COORD

In this section we modify A_1-COORD so as to achieve better PoA when C is relatively large compared to m.

> A_2-COORD: Let N be the assignment. Suppose that job $i \in N_j$. Then the completion of job i is set as $P(i, N_j) = (\phi_{ij})^{\frac{1}{q}}(L(N_j^{\pi(i)-1}) + Cp_{ij})$, for some $0 < q \leq 1$ and $1/q$ is an integer.

Lemma 3.6.1. *The schedule decided by A_2-COORD is feasible.*

Proof. Suppose that in assignment N player k puts jobs i_1, i_2, $\cdots$, $i_{x \leq C}$ on machine j, where the jobs are indexed by their non-decreasing completion times. We need to argue that the completion time $P(i_y, N_j)$ of job i_y, $y \leq x$, is at least as large as the total load of jobs finishing no later than i_y. In the case that multiple jobs among i_1, i_2, $\cdots$, $i_{x \leq C}$ finish at the same time as i_y, W.L.O.G., we can assume that i_y has the largest index. Consider two possibilities.

1. Suppose that none of the jobs belonging to player $\mathcal{P}\backslash\{1, 2, \cdots, k\}$ finishes as early as i_y. Let i_z be the heaviest job among i_1, i_2, $\cdots$, i_y. Then as $P(i_y, N_j) \geq P(i_z, N_j)$,

$$P(i_y, N_j) = (\phi_{i_y j})^{1/q}(L(N_j^{k-1}) + Cp_{i_y j}) \geq$$
$$(\phi_{i_z j})^{1/q}(L(N_j^{k-1}) + Cp_{i_z j}) \geq L(N_j^{k-1}) + \sum_{t=1}^{y} p_{i_t j}.$$

   Observe that the last term in the inequality is at least as large as the total load of the jobs finishing no later than i_y.

2. Suppose that some players in $\mathcal{P}\backslash\{1, 2, \cdots, k\}$ have jobs on machine j that are finished as early as i_y. Let player $k + s$ be such player with the largest index. Suppose further that player $k + s$ has jobs i_1^*, i_2^*, $\cdots$, i_z^* finish as early as i_y on machine j. W.L.O.G., assume that i_1^* is the job with the largest size among the jobs i_1^*, i_2^*, $\cdots$, i_z^*. Then

$$P(i_y, N_j) \geq P(i_1^*, N_j) = (\phi_{i_1^* j})^{1/q}(L(N_j^{k+s-1}) + Cp_{i_1^* j}) \geq$$
$$L(N_j^{k+s-1}) + \sum_{t=1}^{z} p_{i_t^* j},$$

   which is no less than the total load of jobs finishing as early as i_y. The proof follows.

□

Lemma 3.6.2. *Suppose that N is a PNE under A_2-COORD. Then*
$\max_{j\in\mathcal{J}, i\in N_j} P(i,N_j) \le (\sum_{j\in\mathcal{J}} L(N_j)^{q+1})^{\frac{1}{q+1}} + C\max_{j\in\mathcal{J}} L(O_j)$

Proof. Let i^* be the job with the largest completion time. Suppose that it is assigned to j_1 in assignment N and has inefficiency 1 on machine j_2 (j_2 could be the same as j_1). Either $j_2 \neq j_1$, then the player $\pi(i^*)$ controlling i^* has no incentive to move i^* to j_2; or $j_2 = j_1$. In both cases, we have

$$P(i^*, N_{j_1}) \le L(N_{j_2}^{\pi(i^*)-1}) + Cp_{i^*j_2} \le L(N_{j_2}) + Cp_{i^*j_2} \le \\ (\sum_{j\in\mathcal{J}} L(N_j)^{q+1})^{\frac{1}{q+1}} + C\max_{j\in\mathcal{J}} L(O_j),$$

where the last inequality follows from Proposition 3.4.1 and the fact that in the optimal, one machine would have load at least $p_{i^*j_2}$. The proof follows.

□

Lemma 3.6.3. *Suppose that N is a PNE under A_2-COORD. Then*

$$(\sum_{j\in\mathcal{J}} L(N_j)^{q+1})^{\frac{1}{q+1}} \le \frac{[(3q+2)C^{2q}]^{\frac{1}{q+1}} m^{\frac{1}{q+1}}}{(1-q/2)^{\frac{1}{q+1}}} \max_{j\in\mathcal{J}} L(O_j).$$

Proof. Suppose that job i is assigned to machine j_1 in assignment N and machine j_2 in the optimal assignment O. As N is a PNE, $(\phi_{ij_1})^{1/q}(L(N_{j_1}^{\pi(i)-1}) + Cp_{ij_1}) \le (\phi_{ij_2})^{1/q}(L(N_{j_2}^{\pi(i)-1}) + Cp_{ij_2})$. Raising both sides by the power of q, and multiplying them by $p_{i,\min}$, we have

$$p_{ij_1}(L(N_{j_1}^{\pi(i)-1}) + Cp_{ij_1})^q \le p_{ij_2}(L(N_{j_2}^{\pi(i)-1}) + Cp_{ij_2})^q$$

Using the above inequality, we derive

$$p_{ij_1}(L(N_{j_1}^{\pi(i)-1}) + p_{ij_1})^q \le p_{ij_1}(L(N_{j_1}^{\pi(i)-1}) + Cp_{ij_1})^q \le \\ p_{ij_2}(L(N_{j_2}^{\pi(i)-1}) + Cp_{ij_2})^q \le p_{ij_2}(L(N_{j_2}) + Cp_{ij_2})^q.$$

Define $x_{ij} = 1(0)$ if job i is (not) assigned to machine j in assignment N; similarly define $y_{ij} = 1(0)$ if job i is (not) assigned to machine j in the optimal O. Then the above inequality, if summed over all jobs, can be expressed as

$$\begin{aligned}
& \sum_{i\in\mathfrak{I}}\sum_{j\in\mathfrak{J}} x_{ij}p_{ij}(L(N_j^{\pi(i)-1})+x_{ij}p_{ij})^q \\
\leq & \sum_{i\in\mathfrak{I}}\sum_{j\in\mathfrak{J}} y_{ij}p_{ij}(L(N_j)+Cy_{ij}p_{ij})^q \\
\leq & \sum_{i\in\mathfrak{I}}\sum_{j\in\mathfrak{J}} y_{ij}p_{ij}L(N_j)^q + y_{ij}p_{ij}(y_{ij}p_{ij}C)^q \\
= & \sum_{j\in\mathfrak{J}} L(N_j)^q \sum_{i:i\in O_j} p_{ij} + \sum_{j\in\mathfrak{J}} C^q \sum_{i:i\in O_j} p_{ij}^{1+q} \\
\leq & \sum_{j\in\mathfrak{J}} L(N_j)^q L(O_j) + \sum_{j\in\mathfrak{J}} C^q L(O_j)^{1+q} \\
\leq & \sum_{j\in\mathfrak{J}} \frac{(1/q-1)L(O_j)^{1+q} + 2C^q L(O_j)^{1+q} + \frac{1}{2C^q}L(N_j)^{1+q}}{1/q+1} + \sum_{j\in\mathfrak{J}} C^q L(O_j)^{1+q} \\
= & \sum_{j\in\mathfrak{J}} \frac{L(N_j)^{1+q}}{2C^q(1+1/q)} + L(O_j)^{1+q}[C^q(1+\frac{2}{1+1/q})+\frac{1-q}{1+q}], \qquad (3.5)
\end{aligned}$$

where the second inequality follows from Proposition 3.4.3 by setting $p = 1/q$, $a = L(N_j^{\pi(i)-1})$ and $b = Cy_{ij}p_{ij}$; the third inequality from Proposition 3.4.4 by setting $A = 0$, $B_i = p_{ij}$ and $p = 1+q$ (so that $\sum_{i:i\in O_j} p_{ij}^{1+q} \leq (\sum_{i:i\in O_j} p_{ij})^{1+q} = L(O_j)^{1+q}$); and the fourth one from the arithmetic-geometric mean inequality (of $1/q+1$ terms).

We now bound the term $\sum_{j\in\mathfrak{J}} L(N_j)^{q+1}$.

$$\begin{aligned}
\sum_{j\in\mathfrak{J}} L(N_j)^{q+1} & \leq (1+q)C^q \sum_{i\in\mathfrak{I}}\sum_{j\in\mathfrak{J}} x_{ij}p_{ij}(L(N_j^{\pi(i)-1})+x_{ij}p_{ij})^q \\
& \leq (1+q)C^q\{\frac{\sum_{j\in\mathfrak{J}} L(N_j)^{1+q}}{2C^q(1/q+1)} + L(O_j)^{1+q}[C^q(1+\frac{2}{1+1/q})+\frac{1-q}{1+q}]\} \\
& = q/2\sum_{j\in\mathfrak{J}} L(N_j)^{1+q} + (C^{2q}((1+q)+2q)+C^q(1-q))\sum_{j\in\mathfrak{J}} L(O_j)^{1+q} \\
& \leq q/2\sum_{j\in\mathfrak{J}} L(N_j)^{1+q} + C^{2q}(3q+2)\sum_{j\in\mathfrak{J}} L(O_j)^{1+q},
\end{aligned}$$

where the first inequality follows from the same derivation as in (3.3); the second from (3.5); the third from the fact that $C^q(1-q) \leq C^{2q}$.

Rearranging terms in the above inequality, and raising both sides to the power of $1/(1+q)$,

$$(1-q/2)^{\frac{1}{1+q}}(\sum_{j\in\mathcal{J}} L(N_j)^{1+q})^{\frac{1}{q+1}} \le (C^{2q}(3q+2))^{\frac{1}{1+q}}(\sum_{j\in\mathcal{J}} L(O_j)^{1+q})^{\frac{1}{1+q}} \le$$
$$(C^{2q}(3q+2))^{\frac{1}{1+q}} m^{\frac{1}{1+q}} \max_{j\in\mathcal{J}} L(O_j),$$

where the second inequality follows from Proposition 3.4.1. The proof follows.

□

Theorem 3.6.4. *A_2-COORD guarantees a PNE. Such a PNE can be computed in polynomial time and each player can compute her optimal strategy in polynomial time. Moreover, for any fixed q, $0 < q \le 1$ so that $1/q$ is an integer, it guarantees the PoA of $O(C^{\frac{2q}{q+1}} m^{\frac{1}{q+1}})$.*

Proof. The existence of the PNE, and the poly-time computability of such PNEs and of a player's optimal strategy follow the same arguments as in the proof of Theorem 3.5.4. The last part of the theorem is due to Lemmas 3.6.2 and 3.6.3.

□

3.7 A_3-COORD

In this section, we assume that jobs are indexed in such a way that each player controls jobs with consecutive indices. Precisely, let c_k be the number of jobs controlled by player k. Then her jobs are indexed as $1+\sum_{t=1}^{k-1} c_t$, $2+\sum_{t=1}^{k-1} c_t$, $\cdots$, $c_k+\sum_{t=1}^{k-1} c_t$. Unlike the previous two sections, here N_j^i denotes the set of jobs with index at most i that are assigned to machine j in N. Accordingly, $L(N_j^i)$ is their total load: $L(N_j^i) = \sum_{i':i'\le i, i'\in N_j} p_{i'j}$. Let $W = \frac{\max_{i\in\mathcal{I}} \min_{j\in\mathcal{J}} p_{ij}}{\min_{i\in\mathcal{I}} \min_{j\in\mathcal{J}} p_{ij}}$. We assume that all job sizes are rescaled so that $\min_{i\in\mathcal{I}} \min_{j\in\mathcal{J}} p_{ij} = 1$. Then $p_{ij} \ge 1$ for all i, j.

We now introduce A_3-COORD.

> A_3-COORD: Let N be the assignment. Suppose that job $i \in N_j$. Then the completion of job i is set as $P(i, N_j) = (p_{ij})^{\frac{1}{q}}(L(N_j^{i-1})+p_{ij})$. When $C > 1$, we set $q = 1$. When $C = 1$, we set $q = \theta(\log mW)$.

Unlike A_1-COORD and A_2-COORD, here C is absent in the completion time $P(i, N_j)$. Also notice that we replace the inefficiency ϕ_{ij} with the size p_{ij}, hence A_3-COORD is strongly local.

Lemma 3.7.1. *The schedule decided by A_3-COORD is feasible.*

Proof. Let $i_1, i_2, \cdots, i_x$ be the set of jobs assigned to j in N, assuming that their finishing times are non-decreasing. We argue that when job $i_{y \le x}$ finishes, the total load of jobs $i_{y'}$, $y' \le y$, is no larger than $P(i_y, N_j)$. W.L.O.G., if multiple jobs finish at the same time with i_y, then i_y is the one with the largest index.

Suppose that all jobs $i_{y'}$, $y' < y$, have indices smaller than i_y, then $P(i_y, N_j) = (p_{i_y j})^{1/q}(L(N_j^{i_y-1}) + p_{i_y j}) \ge L(N_j^{i_y})$, which is at least as large as the total load of jobs $i_{y'}$, for all $y' \le y$. On the other hand, suppose that a job $i_{y'}$, $y' < y$, has index larger than i_y. W.L.O.G., let $i_{y'}$ be such job with the largest index. Then $P(i_y, N_j) \ge P(i_{y'}, N_j) \ge L(N_j^{i_{y'}}) > \sum_{t=1}^{y} p_{i_t j}$ and the proof follows.

$\square$

Lemma 3.7.2. *Suppose that N is a PNE under A_3-COORD. Suppose that $q = 1$ (thus $C > 1$). Then given any $\gamma \in \mathbb{Z}^+$,*

$$\max_{j \in \mathcal{J}, i \in N_j} P(i, N_j) \le \min \left\{ \begin{array}{c} W(\sqrt{\sum_{j \in \mathcal{J}} L(N_j)^2} + \max_{j \in \mathcal{J}} L(O_j)), \\ \frac{\gamma}{\gamma+1}(\sqrt{\sum_{j \in \mathcal{J}} L(N_j)^2})^{\frac{\gamma+1}{\gamma}} + (W + \frac{W\gamma}{\gamma+1}) \max_{j \in \mathcal{J}} L(O_j). \end{array} \right\}.$$

Suppose that $q = \Theta(\log mW)$ (thus $C = 1$). Then

$$\max_{j \in \mathcal{J}, i \in N_j} P(i, N_j) \le W^{1/q}[(\sum_{j \in \mathcal{J}} L(N_j)^{q+1})^{\frac{1}{q+1}} + \max_{j \in \mathcal{J}} L(O_j)].$$

Proof. Let i^* be the job with the largest completion time in assignment N. Suppose that it is controlled by player $\pi(i^*)$, is assigned to j_1 in assignment N, and has the least size on machine j_2 (j_2 could be the same as j_1). Below we only prove the case of $q = 1$.

Let I_x denote the union of job i^* and the set of jobs controlled by player $\pi(i^*)$ that are assigned to j_2 in N. First assume that $j_2 \ne j_1$. As player $\pi(i^*)$ has no incentive to move i^* to j_2,

$$\begin{aligned} P(i^*, N_{j_1}) + \sum_{i' \in I_x \setminus \{i^*\}} P(i', N_{j_2}) &\le p_{i^* j_2}(L(N_{j_2}^{i^*-1}) + p_{i^* j_2}) \\ &+ \sum_{i' \in I_x, i' < i^*} P(i', N_{j_2}) + \sum_{i' \in I_x, i' > i^*} [P(i', N_{j_2}) + p_{i' j_2} p_{i^* j_2}]. \end{aligned}$$

Notice that the RHS of the inequality is the sum of the costs of the jobs in set I_x if i^* moved to j_2. Observe that the above inequality holds as well when $j_2 = j_1$. Canceling the term $\sum_{i' \in I_x \setminus \{i^*\}} P(i', N_{j_2})$ from both sides of the inequality, we obtain

$$\begin{aligned} P(i^*, N_{j_1}) &\le p_{i^* j_2}(L(N_{j_2}^{i^*-1}) + \sum_{i' \in I_x, i' \ge i^*} p_{i' j_2}) \\ &\le p_{i^* j_2}(L(N_{j_2}) + p_{i^* j_2}) \le WL(N_{j_2}) + W^2. \end{aligned} \tag{3.6}$$

We can further bound the expression $WL(N_{j_2}) + W^2$ in two different ways. First, note that

$$WL(N_{j_2})+W^2 \le W(\sqrt{\sum_{j\in\mathcal{J}} L(N_j)^2}+W) \le W(\sqrt{\sum_{j\in\mathcal{J}} L(N_j)^2}+\max_{j\in\mathcal{J}} L(O_j)), \tag{3.7}$$

where the first inequality follows from Proposition 3.4.1 and the second by the fact that one machine would have load at least W in the optimal. A second way to bound $WL(N_{j_2}) + W^2$ is as follows.

$$\begin{aligned} WL(N_{j_2}) + W^2 &\le \frac{W^{\gamma+1} + \sum_{t=1}^{\gamma}(L(N_{j_2})^{\frac{1}{\gamma}})^{\gamma+1}}{\gamma+1} + W^2 \\ &= \frac{\gamma}{\gamma+1} L(N_{j_2})^{\frac{\gamma+1}{\gamma}} + (W + \frac{W^\gamma}{\gamma+1})W \\ &\le \frac{\gamma}{\gamma+1}(\sqrt{\sum_{j\in\mathcal{J}} L(N_j)^2})^{\frac{\gamma+1}{\gamma}} + (W+\frac{W^\gamma}{\gamma+1}) \max_{j\in\mathcal{J}} L(O_j), \end{aligned} \tag{3.8}$$

where the first inequality follows from the arithmetic-geometric mean inequality (of $\gamma+1$ terms), and the second inequality from the same reason as in (3.7). The inequalities (3.6), (3.7), and (3.8) together give the first part of lemma.

When $q = \Theta(\log mW)$, either $j_2 \neq j_1$, then the player $\pi(i^*)$ controlling i^* has no incentive to move i^* to j_2; or $j_2 = j_1$. In both cases,

$$\begin{aligned} P(i^*, N_{j_1}) \le (p_{i^* j_2})^{1/q}(L(N_{j_2}^{i^*-1}) + p_{i^* j_2}) \le W^{1/q}(L(N_{j_2}) + W) \le \\ W^{1/q}[(\sum_{j\in\mathcal{J}} L(N_j)^{q+1})^{\frac{1}{q+1}} + \max_{j\in\mathcal{J}} L(O_j)], \end{aligned}$$

where the last inequality follows from the same reason as in (3.7). The last part of the lemma follows.

□

In the next two lemmas, we show how to bound the term $(\sum_{j\in\mathcal{J}} L(N_j)^{q+1})^{\frac{1}{q+1}}$, when $q = 1$ and when $q = \Theta(\log mW)$, separately.

Lemma 3.7.3. *Suppose that N is a PNE under A_3-COORD and $q = 1$ (thus $C > 1$). Then*

$$\sqrt{\sum_{j\in\mathcal{J}} L(N_j)^2} \leq \frac{\sqrt{m}}{\sqrt{3/2}-1} \max_{j\in\mathcal{J}} L(O_j).$$

Proof. Let $x_{ij} = 1(0)$ if job i is (not) assigned to machine j in assignment N. Caragiannis ([11], Theorem 7) showed the following inequality.

$$\sum_j \frac{L(N_j)^2}{2} \leq \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} x_{ij}p_{ij}(L(N_j^{i-1}) + x_{ij}p_{ij}). \tag{3.9}$$

The RHS of the inequality is exactly the sum of costs of all players in $\mathcal{P}$ in assignment N. Therefore,

$$\sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} x_{ij}p_{ij}(L(N_j^{i-1}) + x_{ij}p_{ij}) = \sum_{k\in\mathcal{P}} \sum_{i:\pi(i)=k,j:i\in N_j} P(i, N_j). \tag{3.10}$$

Consider player $k \in \mathcal{P}$. She has no incentive to re-assign her jobs to the machines where they belong to in the optimal. Therefore,

$$\sum_{i:\pi(i)=k,j:i\in N_j} P(i, N_j) \leq$$

$$\sum_{j\in\mathcal{J}}[\sum_{i:\pi(i)=k,i\in O_j} p_{ij}L(N_j) + \frac{(\sum_{i:\pi(i)=k,i\in O_j} p_{ij})^2 + \sum_{i:\pi(i)=k,i\in O_j} w_{ij}^2}{2}] \leq$$

$$\sum_{j\in\mathcal{J}}[\sum_{i:\pi(i)=k,i\in O_j} p_{ij}L(N_j) + (\sum_{i:\pi(i)=k,i\in O_j} p_{ij})^2],$$

where the first inequality is derived by assuming (pessimistically) that all jobs of player k in O_j have indices larger than the jobs of all other players in N_j. Summing the above inequality over all players,

$$
\begin{aligned}
\sum_{k\in\mathcal{P}} \sum_{i:\pi(i)=k, j: i\in N_j} P(i, N_j) &\leq \sum_{j\in\mathcal{J}} [L(O_j)L(N_j) + \sum_{k\in\mathcal{P}} (\sum_{i:\pi(i)=k, i\in O_j} p_{ij})^2] \\
&\leq \sum_{j\in\mathcal{J}} L(O_j)L(N_j) + L(O_j)^2 \\
&\leq \sum_{j\in\mathcal{J}} (L(N_j)+L(O_j))^2 - \sum_{j\in\mathcal{J}} L(N_j)^2 \\
&\leq (\sqrt{\sum_{j\in\mathcal{J}} L(N_j)^2} + \sqrt{\sum_{j\in\mathcal{J}} L(O_j)^2})^2 - \sum_{j\in\mathcal{J}} L(N_j)^2 \qquad (3.11)
\end{aligned}
$$

where the second inequality follows from Proposition 3.4.4 (by setting $A = 0$ and $p = 1$), and the last inequality from Minkowski inequality. By (3.9), (3.10), (3.11), and re-arranging terms, we have

$$
(\sqrt{3/2} - 1)\sqrt{\sum_{j\in\mathcal{J}} L(N_j)^2} \leq \sqrt{\sum_{j\in\mathcal{J}} L(O_j)^2} \leq \sqrt{m} \max_{j\in\mathcal{J}} L(O_j),
$$

where the last inequality follows from Proposition 3.4.1. The proof follows. □

Lemma 3.7.4. *Suppose that N is a PNE under A_3-COORD and $q = \Theta(\log mW)$ (thus $C = 1$). Then*

$$
(\sum_{j\in\mathcal{J}} L(N_j)^{q+1})^{\frac{1}{q+1}} \leq e(q+1)m^{\frac{1}{q+1}} \max_{j\in\mathcal{J}} L(O_j).
$$

Proof. Let $x_{ij} = 1(0)$ if job i is (not) assigned to machine j in assignment N; similarly let $y_{ij} = 1(0)$ if job i is (not) assigned to machine j in the optimal O. We make the following claim.

$$
\sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} x_{ij}p_{ij}(L(N_j^{i-1}) + x_{ij}p_{ij})^q \leq \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} y_{ij}p_{ij}(L(N_j) + y_{ij}p_{ij})^q. \qquad (3.12)
$$

By our index scheme and the fact that $C = 1$, job i is controlled by player i. Suppose job i is in j_1 in N and in j_2 in O. As player i has no incentive to move her job from j_1 to j_2,

$$
(p_{ij_1})^{1/q}(L(N_{j_1}^{i-1}) + p_{ij_1}) \leq (p_{ij_2})^{1/q}(L(N_{j_2}^{i-1}) + p_{ij_2}) \leq (p_{ij_2})^{1/q}(L(N_{j_2}) + p_{ij_2}).
$$

Raising the above inequality to the power of q and summing it over all players, we have

$$\sum_{j\in\mathcal{J}}\sum_{i\in N_j} p_{ij}(L(N_j^{i-1})+p_{ij})^q \leq \sum_{j\in\mathcal{J}}\sum_{i\in O_j} p_{ij}(L(N_j)+p_{ij})^q,$$

and Inequality (3.12) follows.

The rest of the analysis is completely the same as Caragiannis (11, Theorem 7). He showed the following two inequalities:

$$(e-1)(q+1)\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} y_{ij}p_{ij}(L(N_j)+y_{ij}p_{ij})^q \leq$$
$$((\sum_{j\in\mathcal{J}} L(N_j)^{q+1})^{\frac{1}{q+1}} + e(\sum_{j\in\mathcal{J}} L(O_j)^{q+1})^{\frac{1}{q+1}})^{q+1} - \sum_{j\in\mathcal{J}} L(N_j)^{q+1} \tag{3.13}$$

$$(e-1)(\sum_{j\in\mathcal{J}} L(N_j))^{q+1} \leq (e-1)(q+1)\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} x_{ij}p_{ij}(L(N_j^{i-1})+x_{ij}p_{ij})^q \tag{3.14}$$

Combining (3.12), (3.13), and (3.14), we have

$$(\sum_{j\in\mathcal{J}} L(N_j)^{q+1})^{\frac{1}{q+1}} \leq \frac{e}{e^{\frac{1}{q+1}}-1}(\sum_{j\in\mathcal{J}} L(O_j)^{q+1})^{\frac{1}{q+1}} \quad\leq\quad e(q+1)m^{\frac{1}{q+1}}\max_{j\in\mathcal{J}} L(O_j),$$

where the second inequality is due to the inequality $e^z - 1 \geq z$ and Proposition 3.4.1. The proof follows.

□

Theorem 3.7.5. *A_3-COORD guarantees a PNE. Moreover, by setting $q=1$, it guarantees the PoA of $O(\min\{W\sqrt{m}, \min_{\gamma\in\mathbb{Z}_{\geq 1}}\{m^{\frac{\gamma+1}{2\gamma}} + W^{\gamma}\}\})$. In case that $C=1$, by setting $q=\theta(\log mW)$, it guarantees the PoA of $O(\log m + \log W)$.*

Proof. The existence of PNE follows the same argument as in the proof of Theorem 3.5.4. The second part of the theorem follows from Lemmas 3.7.2, 3.7.3, and 3.7.4. □

Unfortunately, under A_3-COORD, it is NP-hard to decide the optimal strategy for a player. So we cannot use the same procedure as in the previous two mechanisms to build a PNE in polynomial time. The NP-hardness follows from the observation that when $q=1$, a player controls all the jobs, and only two *identical* machines are given, finding an optimal strategy is equivalent to minimizing the weighted sum of completion times of jobs. (The latter problem is NP-hard by a reduction from the PARTITION problem (29).)

3.8 Counter-examples and Lower Bounds for Known Mechanisms

3.8.1 PNE Existence for ShortestFirst, AJM-2, and BALANCE

Theorem 3.8.1. *ShortestFirst, AJM-2, and BALANCE do not induce PNE even for two identical machines.*

Proof. Consider the instance of two identical machines and two players. Player 1 controls a job of size 1 and a job of size 3. Player 2 controls a job of size 2 and a job of size 4. It is easy to verify that this instance has no PNE when using ShortestFirst. The theorem holds for AJM-2 as well because it is exactly equivalent to ShortestFirst in the case of two identical machines. The same example can be used as an evidence that BALANCE does not induce PNE. □

3.8.2 PNE Existence for Makespan, BCOORD, and CCOORD

Theorem 3.8.2. *Makespan, BCOORD when p=1, and CCOORD when p=1, do not induce PNE even for two identical machines.*

Proof. Consider the instance of two identical machines and two players. Player 1 controls a job of size 1. Player 2 controls a job of size 2, a job of size 1, and a job of size 1/10. It is easy to verify that this instance has no PNE when using the Makespan policy. Notice that when p=1, both BCOORD and CCOORD are equivalent to Makespan in identical machines. □

3.8.3 PoA for ACOORD, BCOORD, and CCOORD

In the following, if we do not specify the size p_{ij} of job i on machine j, we implicitly assume it is infinity. We first present a simple construction where each player controls just two jobs.

Theorem 3.8.3. *For all values of $p \geq 1$, when $C = 2$, ACOORD and BCOORD have PoA of at least $\Omega(m)$. The same bound holds for CCOORD when p=1.*

Proof. Consider an instance of m machines and $n = m - 1$ players. Player k, for $1 \leq k \leq n$, controls two jobs, i_k^0 and i_k^1, both of which can be processed only on machine 0 and machine k. Job i_k^0 has size 1 on machine 0, and size 2 on machine k. Job i_k^1 has size δ on machine 0, and size $\delta \cdot n^p$ on machine k, for some small $\delta > 0$ where $\delta << 1/n^p$.

We first focus on ACOORD. Let the total order of all jobs be as follows $i_1^0, i_1^1, i_2^0, i_2^1, \cdots, i_n^0, i_n^1$. Let N be the following assignment. Each player k assigns job i_k^0 to machine 0 and job i_k^1 to machine k. To see that N is a PNE, consider player k, whose current cost is $k + \delta \cdot n^p$. She has three other possible strategies:

1. Assign both of her jobs to machine 0. Then her cost is $k + k + \delta$.

2. Assign both of her jobs to machine k. Then her cost is $2 \cdot 2^{1/p} + (2 + \delta \cdot n^p) \cdot n$.

3. Assign i_k^0 to machine k and i_k^1 to machine 0. Then her cost is $k - 1 + \delta + 2 \cdot 2^{1/p}$.

All these three strategies incur higher costs, so player k has no reason to deviate. Thus, N is a PNE with makespan of $m - 1$, while OPT is 2.

Next we consider BCOORD (and notice that CCOORD is the same as BCOORD when p=1). We claim that the same assignment N is also a PNE. Consider player k, whose current cost is $n + \delta \cdot n^p$. She has three other possible strategies:

1. Assign both of her jobs to machine 0. Then her cost is $2n + 2\delta$.

2. Assign both of her jobs to machine k. Then her cost is $(2 + \delta \cdot n^p) \cdot 2^{1/p} + (2 + \delta \cdot n^p) \cdot n$.

3. Assign i_k^0 to machine k and i_k^1 to machine 0. Then her cost is $n - 1 + \delta + 2 \cdot 2^{1/p}$.

All these three strategies incur higher costs, so player k has no reason to deviate. Thus, N is a PNE with makespan of $m - 1$, while OPT is 2.

□

We next expand on the same idea to show that the PoA of ACOORD/BCOORD/CCOORD can be much higher when C is large.

Theorem 3.8.4. *For all values of $p \geq 1$, ACOORD and BCOORD have PoA of at least $\Omega(C^{(1-\epsilon)(p+1)} m/p^2)$, for some small $\epsilon > 0$. The same bound holds for CCOORD when p=1.*

Proof. Consider an instance of $m = (n + 1) \cdot (p^2 + \tau + 1)$ machines, where τ is some positive integer (the larger the τ, the smaller the ϵ in the lower bound). Let m_{uv} represent a machine for some u, v, where $0 \leq u \leq (p^2 + \tau)$, $0 \leq v \leq n$. Let k_{st} denote a player for some s, t, where $0 \leq s \leq (p^2 + \tau)$, $1 \leq t \leq n$. We next specify the jobs controlled by each player and their sizes on the machines. Below we assume δ to be some small constant and $\delta << 1/(nC)^{6p^2}$.

1. For player k_{0t}, $1 \leq t \leq n$, she controls two jobs i_{0t}^0, i_{0t}^1. Job i_{0t}^0 has size 1 on machine m_{00}, and size 2 on machine m_{0t}; job i_{0t}^1 has size δ on machine m_{00}, and size $\delta \cdot n^p$ on machine m_{0t}.

2. For player k_{st}, with $s > 0$, $1 \leq t \leq n$, she controls $C+1$ jobs, $i^0_{st}, i^1_{st}, \cdots, i^C_{st}$. Job i^0_{st} has size 1 on machine m_{st}, and size $C^{\sum_{a=1}^{s}(p/(p+1))^a}$ on machine m_{s0}; job i^b_{st}, for $1 \leq b \leq C$, has size δ on machine $m_{(s-1)0}$, and size $\delta(nC)^{2p^2}$ on machine m_{st}.

We first focus on ACOORD. Let the total order for all the jobs be as follows: $i^0_{01}, i^1_{01}, i^0_{02}, i^1_{02}, \cdots, i^0_{0n}, i^1_{0n}, i^0_{11}, i^1_{11}, \cdots, i^C_{11}, i^0_{12}, i^1_{12}, \cdots, i^C_{12}, \cdots, i^0_{1n}, i^1_{1n}, \cdots, i^C_{1n}$, $i^0_{21}, i^1_{21}, \cdots, i^C_{21}, i^0_{22}, i^1_{22}, \cdots, i^C_{22}, \cdots, i^0_{2n}, i^1_{2n}, \cdots, i^C_{2n}, i^0_{31}, i^1_{31}, \cdots, i^C_{31}, i^0_{32}, i^1_{32}, \cdots$, $i^C_{32}, \cdots, i^0_{3n}, i^1_{3n}, \cdots, i^C_{3n}, \cdots, i^0_{(p^2+\tau)1}, i^1_{(p^2+\tau)1}, \cdots, i^C_{(p^2+\tau)1}, i^0_{(p^2+\tau)2}, i^1_{(p^2+\tau)2}, \cdots$, $i^C_{(p^2+\tau)2}, \cdots, i^0_{(p^2+\tau)n}, i^1_{(p^2+\tau)n}, \cdots, i^C_{(p^2+\tau)n}$.

Let N be the following assignment. Each player k_{st} assigns job i^0_{st} to machine m_{s0} and the remaining job(s) to m_{st}. To see that N is a PNE, consider player k_{0t}, $1 \leq t \leq n$. We can use the same argument as in the previous theorem to show that she has no reason to deviate. Next consider player k_{st}, with $s > 0$, $1 \leq t \leq n$, whose current cost is $t \cdot C^{((p+1)/p)\sum_{a=1}^{s}(p/(p+1))^a} + d$, where d is the sum of the cost the jobs i^1_{st}, $i^2_{st}, \cdots, i^C_{st}$, thus $d << 1$. She has three other possible strategies:

1. Assign job i^0_{st} to machine m_{s0} and at least one of the remaining jobs to machine $m_{(s-1)0}$. Then her cost is at least $t \cdot C^{((p+1)/p)\sum_{a=1}^{s}(p/(p+1))^a} + n \cdot C^{\sum_{a=1}^{s-1}(p/(p+1))^a}$.

2. Assign job i^0_{st} and at least one of the remaining jobs to machine m_{st} . Then her cost is at least $1 + (1 + \delta(nC)^{2p^2}) \cdot (nC)^{2p}$.

3. Assign job i^0_{st} to machine m_{st} and the remaining jobs to machine $m_{(s-1)0}$. Then her cost is at least $1+n\cdot C^{1+\sum_{a=1}^{s-1}(p/(p+1))^a} = 1+n\cdot C^{((p+1)/p)\sum_{a=1}^{s}(p/(p+1))^a}$.

All these three strategies incur higher costs, so she has no reason to deviate. Thus, N is a PNE with makespan of $n \cdot C^{((p+1)/p)\sum_{a=1}^{(p^2+\tau)}(p/(p+1))^a} = \Omega(C^{(1-\epsilon)(p+1)}m/p^2)$, for some small $\epsilon > 0$, while OPT is 2.

Next we consider BCOORD (and notice that CCOORD is the same as BCOORD when p=1). We claim that the same assignment N is also a PNE. Consider player k_{0t}, $1 \leq t \leq n$. We can use the same argument as in the previous theorem to show that she has no reason to deviate. Next consider player k_{st}, with $s > 0$, $1 \leq t \leq n$, whose current cost is $n \cdot C^{((p+1)/p)\sum_{a=1}^{s}(p/(p+1))^a} + d$, where d is the sum of the cost the jobs i^1_{st}, $i^2_{st}, \cdots, i^C_{st}$, thus $d << 1$. She has three other possible strategies:

1. Assign job i^0_{st} to machine m_{s0} and at least one of the remaining jobs to machine $m_{(s-1)0}$. Then her cost is at least $n \cdot C^{((p+1)/p)\sum_{a=1}^{s}(p/(p+1))^a} + n \cdot C^{\sum_{a=1}^{s-1}(p/(p+1))^a}$.

2. Assign job i^0_{st} and at least one of the remaining jobs to machine m_{st} . Then her cost is at least $1 + \epsilon(nC)^{2p^2} + (1 + \delta(nC)^{2p^2}) \cdot (nC)^{2p}$.

3. Assign job i^0_{st} to machine m_{st} and the remaining jobs to machine $m_{(s-1)0}$. Then her cost is at least $1+n\cdot C^{1+\sum_{a=1}^{s-1}(p/(p+1))^a} = 1+n\cdot C^{((p+1)/p)+\sum_{a=1}^{s}(p/(p+1))^a}$.

All these three strategies incur higher costs, so she has no reason to deviate. Thus, N is a PNE with makespan of $n \cdot C^{((p+1)/p)\sum_{a=1}^{(p^2+\tau)}(p/(p+1))^a} = \Omega(C^{(1-\epsilon)(p+1)}m/p^2)$, for small $\epsilon > 0$, while OPT is 2.

$\square$

Chapter 4

Weighted Sum of Completion Times Minimization in Multi-Job Games

4.1 Introduction

In this chapter we study *multi-job games*, in which there is a set of agents $\mathcal{A}$ who control arbitrary sets of jobs. Specifically the set of jobs controlled by player $\alpha \in \mathcal{A}$ is denoted by $I(\alpha) \subset \mathcal{J}$ and its cost given a particular schedule is the sum of weighted completion times of its own jobs $\sum_{i \in I(\alpha)} w_i C_i$. As in single-job games, we concentrate on designing coordination mechanisms leading to small coordination ratios, when the social cost is the sum of weighted completion times of all jobs (or equivalently of all agents).

Equilibrium concepts. For the single-job scheduling game the underlying concept of equilibrium is, quite naturally, that of Nash (NE)(39). However, once we allow players to control many jobs and endow them with the weighted completion time cost, already computing a best response to a given situation may be NP-complete. Therefore, it is rather unlikely that such an equilibrium will be attained. To overcome this difficulty we consider a weaker equilibrium concept, which we call *weak equilibrium* (WE), namely, a schedule of all jobs is a WE if no player $\alpha \in \mathcal{A}$ can find a job $i \in I(\alpha)$ such that moving i to a different machine will strictly decrease her cost $\sum_{i \in I(\alpha)} w_i C_i$. We extend the WE concept to mixed (randomized) strategies by allowing player α to keep the distribution of all but one job $i \in \mathcal{J}(\alpha)$ and move job i to any machine. Observe that in the single-job game NE and WE coincide. Throughout, we provide bounds on the coordination ratio of policies for the weak equilibrium, and since NE are also WE our bounds hold for NE as well.

As the reader may have noticed, there is a close connection between WE and local optima of the *jump* (also called *move*) neighborhood (e.g. (53)). In a locally optimal solution of $R||\sum w_i C_i$ for the jump neighborhood, no single job $i \in I$ may be moved to a different machine while decreasing the overall cost. Such solution is exactly a WE when a single player in the scheduling game controls all jobs and the machines use **sr**.

To illustrate the concept of weak equilibrium and the difference between the single-job and the multi-job games consider the following example on 4 machines, $m_1, \ldots, m_4$, with the **sr** policy. There are 4 unit-weight jobs called a, b, c, d such that $p_{am_1} = 1 + \varepsilon$, $p_{bm_1} = 1$, $p_{bm_2} = 1.5$, $p_{cm_2} = 2$, $p_{cm_3} = 3$, $p_{dm_3} = 2$, $p_{dm_4} = 2$, and all other $p_{ij} = +\infty$. In this situation an equilibrium for the single-job game is that jobs a and b go to m_1, job c goes to m_2, and job d goes to m_3, leading to a total cost of $7 + \varepsilon$. Consider now the multi-job game in which one player controls a, b and another player controls c, d. A NE is obtained when a goes to m_1, b goes to m_2, c goes to m_3, and d goes to m_4, and this has total cost $7.5 + \varepsilon$. A WE is obtained from instance when a goes to m_1, b goes to m_2, c goes to m_2, and d goes to m_3, having a total cost of $8 + \varepsilon$.

4.2 Contribution

We start by considering deterministic policies and prove that the coordination ratio of **sr** under WE is exactly 4. This generalizes the result for single-job games (18) and therefore it is the best possible coordination ratio that can be achieved non-preemptively. We prove the upper bound of 4 for **sr** with mixed WE. This is relevant since a pure strategy NE may not exist in this setting (20). Moreover, it is unclear whether the smoothness framework of Roughgarden (45) can be applied here: On the one hand our results hold for the more general framework of WE, while on the other hand having players that control multiple jobs makes it more difficult to prove the (λ, μ)-smoothness.

Before designing improved policies we observe that no anonymous policy may obtain a coordination ratio better than 4, and basically no policy, be it preemptive or randomized, local or strongly local, can achieve a coordination ratio better than 2.618. The latter is in sharp contrast with the case in which players control just one job where better ratios can be achieved with randomized policies (18).

Quite surprisingly we are able to design an "optimal" policy, which we call *externality* (**ex**), that guarantees a coordination ratio of 2.618 for WE. Under this **ex** policy, jobs are processed according to Smith rule but are held back (and not released) for some additional time after completion. This additional time basically equals the negative externality that this particular job imposes over other players. Additionally, we prove that **ex** defines a potential game, so that pure WE exists, and that the convergence time is polynomial. It is worth mentioning that in the single-job game **ex** coincides with the proportional-sharing (**ps**) policy (18), which in turn extends the EQUI policy of the unit-weight case (23). On the other side, when a single player controls all jobs, **ex** coincides with **sr**.

Interestingly, our result for **ex** in case just one player controls all jobs implies a tight approximation guarantee of 2.618 for local optima under the jump neighborhood for $R||\sum w_iC_i$. This tight guarantee also holds for the *swap* neighborhood, in which one is additionally allowed to swap jobs between machines so long as the objective function value decreases (53). In addition, our fast convergence result for **ex** implies another new result, namely, that local search with the jump neighborhood, when only maximum gain steps are taken, converges in polynomial time. These facts appear to be quite surprising since, despite the very large amount of work on local search heuristics for scheduling problems (13; 42), performance guarantees, or polynomial time convergence results are are only known for identical machines (9).

Methodologically our work is based on the inner product framework of (18), but more is needed to deal with the multi-job environment. Our main contribution is however conceptual: One the one hand we demonstrate that the natural economic idea of externalities leads to approximately optimal, and in a way best possible, outcomes even in decentralized systems with only partial information (in a full information and centralized setting one can easily design policies leading to optimal outcomes). One the other hand we provide the first direct application of purely game-theoretic ideas to the analysis of natural and well studied local search heuristics that lead to the currently best known results.

4.3 Preliminaries

Recall that for a player $\alpha \in \mathcal{A}$, the set of job she controls is denoted by $I(\alpha) \subset \mathcal{I}$. Moreover, $\alpha(i)$ denotes the player controlling job i, so that $I(\alpha(i))$ is the set of jobs controlled by who is controlling i.

A *pure strategy profile* is a matrix $\mathbf{x} \in \{0,1\}^{\mathcal{M}\times\mathcal{J}}$ in which $\mathbf{x}_{ij} = 1$ if job i is assigned to machine j. By denoting $\mathbf{x}^\alpha$ the columns of $\mathbf{x}$ corresponding to jobs controlled by player α we say that $\mathbf{x}^\alpha$ is a pure strategy for this player. A mixed strategy for player α is a probability distribution over all $\mathbf{x}^\alpha \in \{0,1\}^{\mathcal{M}\times I(\alpha)}$. A set of mixed strategies for all players $\alpha \in \mathcal{A}$ leads to a (mixed) strategy profile $\mathbf{x} \in [0,1]^{\mathcal{M}\times\mathcal{J}}$ where $\mathbf{x}_{ij}$ is the probability of job i assigned to machine j. Note that the distributions of the different columns of $\mathbf{x}$ may not be independent. We denote by $\mathbf{x}_{-k}$ the matrix obtained by deleting the $k-$th column of $\mathbf{x}$. Observe that $\mathbf{x}_{-k}$ results from the joint probability distribution of all jobs $i' \neq k$ according to $\mathbf{x}$. More precisely $\mathbf{x}_{-k} \in [0,1]^{\mathcal{M}\times\mathcal{J}\setminus\{k\}}$ can be equivalently seen as the mixed strategy profile obtained when players different from $\alpha(k)$ continue using the same strategy, while player $\alpha(k)$ forgets job k and if she was playing the pure strategy $\mathbf{x}^{\alpha(k)} \in \{0,1\}^{\mathcal{M}\times I(\alpha)}$ with probability q, she plays the pure strategy for her jobs different from k, $\mathbf{x}^{\alpha(k)}_{-k} \in \{0,1\}^{\mathcal{M}\times I(\alpha)\setminus\{k\}}$ with probability q (these probabilities add up if she was playing with positive probability two strategies that were equal except for job k). We define $\mathbf{x}_{-K}$ analogously for a set of jobs $K \subseteq \mathcal{J}$.

Given a mechanism $\mathbf{m} \in \{\mathbf{sr}, \mathbf{ex}\}$ and a strategy profile $\mathbf{x}$, $\mathbb{E}[C_i^{\mathbf{m}}(\mathbf{x})]$ is the expected completion time of job i. The conditional expected completion time of job i on machine j when job k is assigned to machine j is denoted $\mathbb{E}[C_i^{\mathbf{m}}(\mathbf{x}_{-k}, k \to j)]$. The expected cost of the strategy profile $\mathbf{x}$ is $\mathbb{E}[C^{\mathbf{m}}(\mathbf{x})] = \sum_{i\in\mathcal{J}} w_i \mathbb{E}[C_i^{\mathbf{m}}(\mathbf{x})]$ and the expected cost of a player α under $\mathbf{x}$ is $\mathbb{E}[C_\alpha^{\mathbf{m}}(\mathbf{x})] = \sum_{i\in I(\alpha)} w_i \mathbb{E}[C_i^{\mathbf{m}}(\mathbf{x})]$. For convenience we also define $\mathbb{E}[C_\alpha^{\mathbf{m}}(\mathbf{x}_{-k}, k \to j)] = \sum_{i\in I(\alpha)} w_i \mathbb{E}[C_i^{\mathbf{m}}(\mathbf{x}_{-k}, k \to j)]$. Note that $\mathbb{E}[C^{\mathbf{m}}(\mathbf{x})] = \sum_{\alpha\in\mathcal{A}} \mathbb{E}[C_\alpha^{\mathbf{m}}(\mathbf{x})]$.

A Nash equilibrium (NE) is therefore a strategy profile $\mathbf{x}$ such that for all player $\alpha \in \mathcal{A}$ and all strategy profiles $\mathbf{y}^\alpha$ for player α we have that:

$$\mathbb{E}[C_\alpha^{\mathbf{m}}(\mathbf{x})] \leq \mathbb{E}[C_\alpha^{\mathbf{m}}(\mathbf{y}^\alpha, \mathbf{x}_{-I(\alpha)})].$$

Similarly, a weak equilibrium (WE) is a strategy profile $\mathbf{x}$ such that for all player $\alpha \in \mathcal{A}$, all jobs $k \in I(\alpha)$, and all machines $i \in \mathcal{M}$, we have that:

$$\mathbb{E}[C_\alpha^{\mathbf{m}}(\mathbf{x})] \leq \mathbb{E}[C_\alpha^{\mathbf{m}}(\mathbf{x}_{-k}, k \to j)].$$

The optimal assignement is the assignment in which the jobs are processed non-preemptively on the machines so that the cost is minimized. Throughout the chapter, $\mathbf{x}^*$ denotes the optimal assignment (thus $\mathbf{x}^*$ is a pure strategy), and we define X_j^* as the set of jobs assigned to machine j under the optimal assignment. Given the assignment of jobs to machines, it is well-known that Smith Rule minimizes the total cost of jobs. Therefore $C^{\mathbf{sr}}(\mathbf{x}^*)$ is the optimal cost.

4.4 Non-preemptive mechanisms

We now study non-preemptive mechanisms (jobs have IDs, needed to break ties between identically looking jobs) and prove that **sr** has a coordination ratio of 4 for mixed WE. We work with mixed strategies since **sr** does not guarantee that existence of pure WE. As mentioned earlier, our result is best possible among non-preemptive mechanisms (18).

Recall that under **sr**, each machine j schedules non-preemptively its assigned jobs i in nondecreasing order of $\rho_{ij} = p_{ij}/w_i$, and ties are broken using the IDs. To simplify the presentation, we say that $\rho_{kj} < \rho_{ij}$ if k comes earlier than i in the **sr** order of machine j. Thus, given a strategy profile $\mathbf{x}$ we have $\mathbb{E}[C_i^{\mathbf{sr}}(\mathbf{x}_{-i}, i \to j)] = p_{ij} + \sum_{k:\rho_{kj}<\rho_{ij}} \mathbf{x}_{kj} p_{kj}$ so that,

$$\begin{aligned}\mathbb{E}\left[C^{\mathbf{sr}}(\mathbf{x})\right] &= \sum_{i\in\mathcal{J}} w_i \sum_{j\in\mathcal{M}} \mathbf{x}_{ij}\mathbb{E}[C_i^{\mathbf{sr}}(\mathbf{x}_{-i}, i \to j)] \\ &= \sum_{j\in\mathcal{M}}\sum_{i\in\mathcal{J}} \mathbf{x}_{ij} w_i (p_{ij} + \sum_{k:\rho_{kj}<\rho_{ij}} \mathbf{x}_{kj} p_{kj}).\end{aligned} \tag{4.1}$$

Extending the inner product space technique of Cole et al. (18), we let $\varphi : \mathbf{x} \to L_2([0,\infty])^{\mathcal{M}}$, which maps every strategy profile $\mathbf{x}$ to a vector of functions (one for each machine) as follows. If $\boldsymbol{f} = \varphi(\mathbf{x})$, then for each $j \in \mathcal{M}$, the j-th component of $\boldsymbol{f}$ is the function $f_j(y) := \sum_{i\in\mathcal{J},\rho_{ij}\geq y} \mathbf{x}_{ij} w_i$. Letting $\langle f_j, g_j\rangle = \int_0^\infty f_j(y) g_j(y) dy$ be the standard inner product on L_2 we get that $\langle \boldsymbol{f}, \mathbf{g}\rangle = \sum_{j\in\mathcal{M}} \langle f_j, g_j\rangle$. Additionally, we let $\eta_j(\mathbf{x}) = \sum_{i\in\mathcal{J}} w_i \mathbf{x}_{ij} p_{ij}$ and $\eta(\mathbf{x}) = \sum_{j\in\mathcal{M}} \eta_j(\mathbf{x})$.

The next lemma and expressions (4.2) and (4.3) follow easily from the derivations of Cole et al. (18). The only difference is that here we need to prove the results for mixed strategies.

Lemma 4.4.1. *For a strategy profile* $\mathbf{x}$ *and the optimal assignment* $\mathbf{x}^*$*, let* $\boldsymbol{f} = \varphi(\mathbf{x})$ *and* $\boldsymbol{f}^* = \varphi(\mathbf{x}^*)$*. Then* $\langle fj, f_j^*\rangle = \sum_{i\in X_j^*} \sum_{k\in\mathcal{J}} w_i w_k \mathbf{x}_{kj} \min\{\rho_{ij}, \rho_{kj}\}$.

Similarly to Lemma 4.4.1, and using equation (4.1), we may evaluate

$$||\varphi(\mathbf{x})||^2 \quad\leq\quad 2\mathbb{E}\left[C^{\mathbf{sr}}(\mathbf{x})\right]. \tag{4.2}$$

Additionally, when $\mathbf{x}$ is a pure strategy we have:

$$C^{\mathbf{sr}}(\mathbf{x}) = \frac{1}{2}||\varphi(\mathbf{x})||^2 + \frac{1}{2}\eta(\mathbf{x}). \tag{4.3}$$

Lemma 4.4.1. By definition $\langle f_j, f_j^* \rangle = \int_0^\infty \sum_{i:\rho_{ij}\geq y} \mathbf{x}_{ij} w_i \sum_{k:\rho_{kj}\geq y} \mathbf{x}^*_{kj} w_k \, dy$, which equals

$$\sum_{i,k\in\mathcal{J}} \mathbf{x}_{ij} w_i \mathbf{x}^*_{kj} w_k \int_0^\infty \mathbf{1}_{\rho_{ij}\geq y} \mathbf{1}_{\rho_{kj}\geq y} dy = \sum_{i,k\in\mathcal{J}} \mathbf{x}_{ij} w_i \mathbf{x}^*_{kj} w_k \min\{\rho_{ij}, \rho_{kj}\}$$
$$= \sum_{i\in X_j^*} \sum_{k\in\mathcal{J}} w_i w_k \mathbf{x}_{kj} \min\{\rho_{ij}, \rho_{kj}\}.$$

Similarly to Lemma 4.4.1, and using equation (4.1), we may evaluate

$$\begin{aligned}
||\varphi(\mathbf{x})||^2 &= \sum_{j\in\mathcal{M}} \int_0^\infty f_j^2(y) dy = \sum_{j\in\mathcal{M}} \sum_{i,k\in\mathcal{J}} w_i \mathbf{x}_{ij} w_k \mathbf{x}_{kj} \min\{\rho_{ij}, \rho_{kj}\} \\
&= \sum_{j\in\mathcal{M}} \sum_{i\in\mathcal{J}} \mathbf{x}_{ij} w_i (2 \sum_{k:\rho_{kj}<\rho_{ij}} \mathbf{x}_{kj} w_k \rho_{kj} + w_i \rho_{ij} \mathbf{x}_{ij}) \\
&= 2\mathbb{E}\left[C^{\mathbf{sr}}(\mathbf{x})\right] - \sum_{j\in\mathcal{M}} \sum_{i\in\mathcal{J}} w_i \mathbf{x}_{ij} (2 - \mathbf{x}_{ij}) w_i \rho_{ij} \\
&= 2\mathbb{E}\left[C^{\mathbf{sr}}(\mathbf{x})\right] - (2\eta(\mathbf{x}) - \sum_{j\in\mathcal{M}} \sum_{i\in\mathcal{J}} w_i \mathbf{x}_{ij}^2 p_{ij}) \\
&\leq 2\mathbb{E}\left[C^{\mathbf{sr}}(\mathbf{x})\right].
\end{aligned}$$

Proving (4.2). Additionally, the previous derivation when $\mathbf{x}$ is a pure strategy leads to

$$C^{\mathbf{sr}}(\mathbf{x}) = \frac{1}{2} ||\varphi(\mathbf{x})||^2 + \frac{1}{2}\eta(\mathbf{x}).$$

□

In what follows, let $\mathbf{x}$ denote a mixed weak equilibrium and $\mathbf{x}^*$ the optimal assignment. Let $\boldsymbol{f} = \varphi(\mathbf{x})$ and $\boldsymbol{f}^* = \varphi(\mathbf{x}^*)$.

Lemma 4.4.2. *Consider $X_j^*(\alpha) = X_j^* \cap I(\alpha)$, the jobs of player α assigned to machine j in the optimal solution. Then for each $i \in X_j^*(\alpha)$ we have:*

$$w_i \mathbb{E}\left[C_i^{\mathbf{sr}}(\mathbf{x})\right] \leq w_i (p_{ij} + \sum_{k:\rho_{kj}<\rho_{ij}} \mathbf{x}_{kj} p_{kj}) + p_{ij} \sum_{k\in I(\alpha)\setminus\{i\}, \rho_{kj}>\rho_{ij}} w_k \mathbf{x}_{kj}.$$

Proof. Define $\overline{\mathbf{x}}$ to be the strategy profile where player $\alpha(i)$ changes the probability distribution of her job i to the optimal assignment while the joint-probability distribution of the rest of the jobs $i' \neq i$ remain unchanged. That is, $\overline{\mathbf{x}}_{ij} = 1$ and $\overline{\mathbf{x}}_{ij'} = 0$ if $j' \neq j$ and $\overline{\mathbf{x}}_{i'j'} = \mathbf{x}_{i'j'}$ if $i' \neq i$. As $\mathbf{x}$ is a WE, $\mathbb{E}[C_\alpha^{\mathbf{sr}}(\mathbf{x})] \leq \mathbb{E}[C_\alpha^{\mathbf{sr}}(\overline{\mathbf{x}})]$, thus the cost of player α, which equals $w_i\mathbb{E}[C_i^{\mathbf{sr}}(\mathbf{x})] + \sum_{k\in I(\alpha)\setminus\{i\}} w_k\mathbb{E}[C_k^{\mathbf{sr}}(\mathbf{x})]$, is upper bounded by

$$w_i(p_{ij} + \sum_{k:\rho_{kj}<\rho_{ij}} \mathbf{x}_{kj}p_{kj}) + \sum_{k\in I(\alpha)\setminus\{i\}} w_k\mathbb{E}[C_k^{\mathbf{sr}}(\mathbf{x})] + w_k\mathbf{x}_{kj}p_{kj},$$

where the first term is the expected cost of job i and the second is the largest possible increase in the expected cost on the other jobs controlled by α, say if i is moved to the machine where k is. The proof follows by canceling $\sum_{k\in I(\alpha)\setminus\{i\}} w_k\mathbb{E}[C_k^{\mathbf{sr}}(\mathbf{x})]$. □

Lemma 4.4.3. *For a machine $j \in \mathcal{M}$, $\sum_{i\in X_j^*} w_i\mathbb{E}[C_i^{\mathbf{sr}}(\mathbf{x})] - \eta_j(\mathbf{x}^*) \leq \langle f_j, f_j^*\rangle$.*

Proof. For a job $i \in X_j^*(\alpha)$ we can apply Lemma 4.4.2 to get

$$\begin{aligned} w_i\mathbb{E}[C_i^{\mathbf{sr}}(\mathbf{x})] - w_ip_{ij} &\leq w_i \sum_{k:\rho_{kj}<\rho_{ij}} \mathbf{x}_{kj}p_{kj} + p_{ij} \sum_{k\in I(\alpha)\setminus\{i\},\rho_{kj}>\rho_{ij}} w_k\mathbf{x}_{kj} \\ &\leq w_i \sum_{k:\rho_{kj}<\rho_{ij}} \mathbf{x}_{kj}w_{kj}\rho_{kj} + w_i\rho_{ij} \sum_{k:\rho_{kj}>\rho_{ij}} w_k\mathbf{x}_{kj} \\ &\leq \sum_{k\in\mathcal{J}} w_iw_k\mathbf{x}_{kj}\min\{\rho_{ij}\rho_{kj}\}. \end{aligned}$$

Since $X_j^* = \sum_{\alpha\in\mathcal{A}} X_j^*(\alpha)$ and $\eta_j(\mathbf{x}^*) = \sum_{i\in X_j^*} w_ip_{ij}$, and using Lemma 4.4.1

$$\sum_{i\in X_j^*} w_i\mathbb{E}[C_i^{\mathbf{sr}}(\mathbf{x})] - \eta_j(\mathbf{x}^*) \quad\leq\quad \sum_{i\in X_j^*}\sum_{k\in\mathcal{J}} w_iw_k\mathbf{x}_{kj}\min\{\rho_{ij}\rho_{kj}\} = \langle f_j, f_j^*\rangle.$$

□

Theorem 4.4.4. $\mathbb{E}[C^{\mathbf{sr}}(\mathbf{x})] \leq 4C^{\mathbf{sr}}(\mathbf{x}^*)$.

Proof. Using Lemma 4.4.3, we have that $\mathbb{E}[C^{\mathbf{sr}}(\mathbf{x})]$ can be bounded as

$$\sum_{i\in\mathcal{J}} w_i\mathbb{E}[C_i^{\mathbf{sr}}(\mathbf{x})] \leq \sum_{j\in\mathcal{M}}\sum_{i\in X_j^*} w_i\mathbb{E}[C_i^{\mathbf{sr}}(\mathbf{x})] \leq \langle \boldsymbol{f}, \boldsymbol{f}^*\rangle + \eta(\mathbf{x}^*).$$

From Cauchy-Schwartz inequality, equation (4.3) and inequality (4.2) we get that the latter is at most:

$$||\boldsymbol{f}^*||^2 + \frac{1}{4}||\boldsymbol{f}||^2 + \eta(\mathbf{x}^*) = 2C^{\mathbf{sr}}(\mathbf{x}^*) + \frac{1}{4}||\boldsymbol{f}||^2 \leq 2C^{\mathbf{sr}}(\mathbf{x}^*) + \frac{1}{2}\mathbb{E}\left[C^{\mathbf{sr}}(\mathbf{x})\right].$$

Therefore $\mathbb{E}[C^{\mathbf{sr}}(\mathbf{x})] \leq 2C^{\mathbf{sr}}(\mathbf{x}^*) + \frac{1}{2}\mathbb{E}\left[C^{\mathbf{sr}}(\mathbf{x})\right]$, and the result follows. □

4.5 Preemptive mechanisms

Finding policies that beat the coordination ratio of 4 for WE is impossible if we restrict to non-preemptive ones. This holds even for the single-job game (18), where WE and NE coincide. Therefore we need to consider preemptive or randomized policies. In Section 4.8 we first observe that even with preemption, if we restrict to anonymous policies, beating the ratio of 4 is not possible. Furthermore, we prove that the absolute limit for basically any policy, be it preemptive or randomized, using even global information, and even if different machines use different policies, is $1 + \phi \approx 2.618$, where ϕ is the golden ratio. The precise set of policies for which this lower bound holds are those such that when machine $j \in M$ is assigned a single job, $i \in I$, then $C_i = p_{ij}$.

As the performance of **sr** coincides in the single-job and multi-job games one may wonder whether natural preemptive policies, that work well in the single-job game, also do in the multi-job game. Unfortunately this is not the case as we show in Section 4.8. Indeed we prove that the champion preemptive policy for the single-job game, Proportional-sharing (18; 23), has a coordination ratio of at least 5.848 for WE and at least 2.848 for NE. It is thus rather surprising that we can actually achieve this ratio with a fairly natural policy, externality (**ex**). A key ingredient of this policy is that it heavily relies on the ownership of the jobs, a feature that policies for the single-job game certainly do not share.

The results in this section are presented for pure strategy profiles. This is primarily done for simplicity and also because, as we will show later, our preemptive policy induces a potential game and therefore pure WE are guaranteed to exist. Thus, given a pure strategy profile $\mathbf{x}$, we may refer to $\mathbf{x}$ as an assignment, and we may let X_j denote the set of jobs assigned to machine j under $\mathbf{x}$, i.e., $i \in X_j$ if $\mathbf{x}_{ij} = 1$. Let also $X_j(\alpha) = X_j \cap I(\alpha)$ be the set of jobs controlled by player α on this machine j under $\mathbf{x}$.

Recall that in the proportional sharing policy (**ps**) (18), the machine processing power is split among the assigned jobs proportionally to their weight. Given an assignment $\mathbf{x}$, if job i is assigned to machine j, it can be observed that:

$$C_i^{\mathbf{ps}}(\mathbf{x}) = p_{ij} + \sum_{k \in X_j, \rho_{kj} < \rho_{ij}} p_{kj} + p_{ij} \sum_{k \in X_j \setminus \{i\}, \rho_{kj} > \rho_{ij}} \frac{w_k}{w_i}.$$

Proposition 4.5.1 ((18)). *Given an assignment* $\mathbf{x}$*,* $C^{\mathbf{ps}}(\mathbf{x}) = ||\varphi(\mathbf{x})||^2$.

In our externality policy, **ex**, given an assignment $\mathbf{x}$, the machine processes the jobs according to **sr** but once a job is completed, it is delayed for an amount of time accounting for the negative externality it is imposing on other players. Thus in **ex** the cost for the owner of job j due to this job will be

$$w_i C_i^{\mathbf{ex}}(\mathbf{x}) = w_i p_{ij} + w_i \sum_{k \in X_j, \rho_{kj} < \rho_{ij}} p_{kj} + p_{ij} \sum_{k \in X_j \setminus I(\alpha(i)), \rho_{kj} > \rho_{ij}} w_k.$$

The completion time of i is then defined by the previous equation. Observe that in the single-job game, **ex** reduces to **ps**, while if all jobs are controlled by a single player **ex** reduces to **sr**. Also, **ex** induces feasible schedules since no completion time can be smaller than that given by Smith-rule. Policy **ex** can be seen as a preemptive policy in which jobs are processed as in **sr**, except for an infinitesimal piece that is processed at the time defined by previous equation. Moreover **ex** is strongly local and nonanonymous. A consequence of the definitions of **sr**, **ps**, and **ex** is that for a fixed assignment $\mathbf{x}$ their costs satisfy:

$$\begin{aligned} C^{\mathbf{ex}}(\mathbf{x}) &= C^{\mathbf{sr}}(\mathbf{x}) + \sum_{j \in \mathcal{M}} \sum_{i \in X_j} p_{ij} \sum_{k \in X_j \setminus I(\alpha(i)), \rho_{kj} > \rho_{ij}} w_k \qquad (4.4) \\ &= C^{\mathbf{ps}}(\mathbf{x}) - \sum_{j \in \mathcal{M}} \sum_{i \in X_j} p_{ij} \sum_{k \in X_j(\alpha(i)), \rho_{kj} > \rho_{ij}} w_k. \end{aligned}$$

In the following, let $\mathbf{x}^*$ be an optimal assignment and $\mathbf{x}$ a WE. We also let $\varphi(\mathbf{x}) = \boldsymbol{f}$ and $\varphi(\mathbf{x}^*) = \boldsymbol{f}^*$ be as in the previous section.

Lemma 4.5.2. *Consider a job $i \in X_j^*$ and assume i is on j' under $\mathbf{x}$. Then*

$$w_i C_i^{\mathbf{ex}}(\mathbf{x}) \leq w_i(p_{ij} + \sum_{\substack{k \in X_j, \\ \rho_{kj} < \rho_{ij}}} p_{kj}) + p_{ij} \sum_{\substack{k \in X_j, \\ \rho_{kj} > \rho_{ij}}} w_k - p_{ij'} \sum_{\substack{k \in X_{j'}(\alpha(i)), \\ \rho_{ki'} > \rho_{ij'}}} w_k.$$

Proof. The case $j' = j$ is immediate. For $j' \neq j$, consider the cost of jobs belonging player $\alpha(i)$ on machines j or j' under $\mathbf{x}$, which is,

$$w_i C_i^{\mathbf{ex}}(\mathbf{x}) + \sum_{k \in ((X_j(\alpha) \cup X_{j'}(\alpha)) \setminus \{i\}} w_k C_k^{\mathbf{ex}}(\mathbf{x}). \qquad (4.5)$$

Suppose that she moves i from machine j' to j, then the total cost of the same set of jobs is

$$\begin{aligned} &\sum_{k \in ((X_j(\alpha) \cup X_{j'}(\alpha)) \setminus \{i\}} w_k C_k^{\mathbf{ex}}(\mathbf{x}) - p_{ij'} \sum_{k \in X_{j'}(\alpha(i)), \rho_{kj'} > \rho_{ij'}} w_k + \\ &w_i(p_{ij} + \sum_{\substack{k \in X_j, \\ \rho_{kj} < \rho_{ij}}} p_{kj}) + p_{ij} \sum_{\substack{k \in X_j \setminus I(\alpha(i)), \\ \rho_{kj} > \rho_{ij}}} w_k + p_{ij} \sum_{\substack{k \in X_j(\alpha(i)), \\ \rho_{kj} > \rho_{ij}}} w_k. \qquad (4.6) \end{aligned}$$

Here the second term is the saving of the cost for those jobs $k \in \alpha(i)$ on machine j' that have larger ratios $\rho_{kj'}$ than $\rho_{ij'}$; the third and fourth terms are the cost of job i on machine j; and the fifth term is the increase of the cost of those jobs $k \in \alpha(i)$ on machine j that have larger ratios ρ_{kj} than ρ_{ij}. As $\mathbf{x}$ is a WE, the term (4.5) is upper bounded by (4.6). $\square$

Lemma 4.5.3. $C^{\mathbf{ex}}(\mathbf{x}) \leq \eta(\mathbf{x}^*) + \langle \boldsymbol{f}, \boldsymbol{f}^* \rangle - \sum_{j \in \mathcal{M}} \sum_{i \in X_j} p_{ij} \sum_{k \in X_j(\alpha(i)), \rho_{kj} > \rho_{ij}} w_k.$

Proof. By Lemma 4.5.2 and summing over all jobs in $\mathcal{J}$, we have that the total cost under $\mathbf{ex}$, $\sum_{i \in \mathcal{J}} w_i C_i^{\mathbf{ex}}(\mathbf{x})$ is upper bounded by

$$\eta(\mathbf{x}^*) + \sum_{j \in \mathcal{M}} \Big(\sum_{i \in X_j^*} w_i \sum_{\substack{k \in X_j, \\ \rho_{kj} < \rho_{ij}}} p_{kj} + \sum_{i \in X_j^*} p_{ij} \sum_{\substack{k \in X_j, \\ \rho_{kj} > \rho_{ij}}} w_k - \sum_{i \in X_j} p_{ij} \sum_{\substack{k \in X_j(\alpha(i)), \\ \rho_{kj} > \rho_{ij}}} w_k \Big). \tag{4.7}$$

By Lemma 4.4.1 and the fact that $\mathbf{x}$ is pure, we have

$$\langle f_j, f_j^* \rangle = \sum_{i \in X_j^*} \sum_{k \in X_j} w_i w_k \min\{\rho_{ij}, \rho_{kj}\} = \sum_{i \in X_j^*} \Big(w_i \sum_{\substack{k \in X_j, \\ \rho_{kj} \leq \rho_{ij}}} p_{kj} + p_{ij} \sum_{\substack{k \in X_j, \\ \rho_{kj} > \rho_{ij}}} w_k \Big).$$

Summing over $j \in \mathcal{M}$ and subtracting the latter from (4.7)

$$C^{\mathbf{ex}}(\mathbf{x}) - \langle \boldsymbol{f}, \boldsymbol{f}^* \rangle \leq \eta(\mathbf{x}^*) - \sum_{j \in \mathcal{M}} \sum_{i \in X_j} p_{ij} \sum_{k \in X_j(\alpha(i)), \rho_{kj} > \rho_{ij}} w_k.$$

$\square$

Theorem 4.5.4. *Let ϕ be the golden ratio. Then $C^{\mathbf{ex}}(\mathbf{x}) \leq (1 + \phi) C^{\mathbf{sr}}(\mathbf{x}^*)$.*

Proof. Lemma 4.5.3 and Cauchy-Schwartz inequality imply that for $\beta > 1/4$

$$\begin{aligned}
C^{\mathbf{ex}}(\mathbf{x}) &\leq \eta(\mathbf{x}^*) + \beta ||\boldsymbol{f}^*||^2 + \frac{1}{4\beta} ||\boldsymbol{f}||^2 - \sum_{j \in \mathcal{M}} \sum_{i \in X_j} p_{ij} \sum_{k \in X_j(\alpha(i)), \rho_{kj} > \rho_{ij}} w_k \\
&\leq \eta(\mathbf{x}^*) + \beta ||\boldsymbol{f}^*||^2 + \frac{1}{4\beta} ||\boldsymbol{f}||^2 - \frac{1}{4\beta} \sum_{j \in \mathcal{M}} \sum_{i \in X_j} p_{ij} \sum_{k \in X_j(\alpha(i)), \rho_{kj} > \rho_{ij}} w_k \\
&\leq \eta(\mathbf{x}^*) + 2\beta C^{\mathbf{sr}}(\mathbf{x}^*) - \beta \eta(\mathbf{x}^*) + \frac{1}{4\beta} C^{\mathbf{ex}}(\mathbf{x}) \\
&\leq (\beta + 1) C^{\mathbf{sr}}(\mathbf{x}^*) + \frac{1}{4\beta} C^{\mathbf{ex}}(\mathbf{x}),
\end{aligned}$$

where the third inequality follows from equation (4.3), from Proposition 4.5.1 and from equation (4.4). By letting $\beta = \frac{1+\sqrt{5}}{4}$ the result follows . $\square$

As mentioned earlier, it turns out that **ex** is best possible. The proof of this fact is deferred to 4.8.

Theorem 4.5.5. *The coordination ratio for weak equilibrium of any prompt mechanism is at least* $1+\phi$.

4.6 Potential function

In this section, we show that **ex** is a potential game and therefore, it guarantees the existence of pure WE.

Theorem 4.6.1. **ex** *induces exact potential games, with the potential*

$$\Phi(\mathbf{x}) = \frac{1}{2}C^{\mathbf{ex}}(\mathbf{x}) + \frac{1}{2}\Gamma(\mathbf{x}),$$

where

$$\Gamma(\mathbf{x}) = \sum_{\alpha\in\mathcal{A}}\sum_{j\in\mathcal{M}}\sum_{i\in X_j(\alpha)} p_{ij} \sum_{k\in X_j(\alpha),\rho_{kj}\geq\rho_{ij}} w_k.$$

Proof. Suppose that player α moves her job i from machine j to j'. Let $\mathbf{x}$ and $\mathbf{x}'$ be the old and new assignments respectively. Observe the change in the cost of player α, namely, $C^{\mathbf{ex}}_{\alpha}(\mathbf{x}') - C^{\mathbf{ex}}_{\alpha}(\mathbf{x})$ is:

$$\begin{aligned}
& w_i\left(\sum_{k\in X_{j'}\cup\{i\},\rho_{kj'}\leq\rho_{ij'}} p_{kj'} + \sum_{k\in X_{j'}\setminus J(\alpha),\rho_{kj'}>\rho_{ij'}} p_{ij'}\frac{w_k}{w_i}\right) + p_{ij'}\sum_{\substack{k\in X_{j'}(\alpha),\\ \rho_{kj'}>\rho_{ij'}}} w_k \\
& -w_i\left(\sum_{k\in X_j\ \rho_{kj}\leq\rho_{ij}} p_{kj} + \sum_{k\in X_j\setminus J(\alpha),\rho_{kj}>\rho_{ij}} p_{ij}\frac{w_k}{w_i}\right) - p_{ij}\sum_{k\in X_j(\alpha),\rho_{kj}>\rho_{ij}} w_k \\
& = w_i\sum_{k\in X_{j'}\cup\{i\}} w_k \min\{\rho_{ij'},\rho_{kj'}\} - w_i\sum_{k\in X_j} w_i\min\{\rho_{ij},\rho_{kj}\}. \qquad (4.8)
\end{aligned}$$

Interestingly, the latter quantity does not depend on α.

Next consider the difference the potential function value of $\mathbf{x}$ and $\mathbf{x}'$, starting from the cost change:

$$\begin{aligned}
C^{\mathbf{ex}}(\mathbf{x}') - C^{\mathbf{ex}}(\mathbf{x}) \;=\;& w_i(\sum_{\substack{k\in X_{j'}\cup\{i\},\\ \rho_{kj'}\le\rho_{ij'}}} p_{kj'} + \sum_{\substack{k\in X_{j'}\setminus J(\alpha),\\ \rho_{kj'}>\rho_{ij'}}} p_{ij'}\frac{w_k}{w_i}) \\
& + p_{ij'}\sum_{\substack{k\in X_{j'},\\ \rho_{kj'}>\rho_{ij'}}} w_k + \sum_{\substack{k\in X_{j'}\setminus J(\alpha),\\ \rho_{kj'}<\rho_{ij'}}} w_k p_{kj'}\frac{w_i}{w_k} \\
& - w_i(\sum_{\substack{k\in X_j\\ \rho_{kj}\le\rho_{ij}}} p_{kj} + \sum_{\substack{k\in X_j\setminus J(\alpha),\\ \rho_{kj}>\rho_{ij}}} p_{ij}\frac{w_k}{w_i}) \\
& - p_{ij}\sum_{\substack{k\in X_j,\\ \rho_{kj}>\rho_{ij}}} w_k - \sum_{\substack{k\in X_j\setminus J(\alpha),\\ \rho_{kj}<\rho_{ij}}} w_k p_{kj}\frac{w_i}{w_k}. \qquad (4.9)
\end{aligned}$$

Finally, the difference in the function Γ can be expressed as

$$\begin{aligned}
\Gamma(\mathbf{x}') - \Gamma(\mathbf{x}) \;=\;& w_i \sum_{k\in X_{j'}(\alpha)\cup\{i\},\rho_{kj'}\le\rho_{ij'}} p_{kj'} + p_{ij'} \sum_{k\in X_{j'}(\alpha),\rho_{kj'}>\rho_{ij'}} w_k \\
& - w_i \sum_{k\in X_j(\alpha),\rho_{kj}\le\rho_{ij}} p_{kj} + p_{ij} \sum_{k\in X_j(\alpha),\rho_{kj}>\rho_{ij}} w_k. \qquad (4.10)
\end{aligned}$$

Combining (4.9) and (4.10) yields

$$\Phi(\mathbf{x}') - \Phi(\mathbf{x}) = w_i \sum_{k\in X_{j'}\cup\{i\}} w_k \min\{\rho_{ij'},\rho_{kj'}\} - w_i \sum_{k\in X_j} w_k \min\{\rho_{ij},\rho_{kj}\},$$

which equals the difference of player α's cost as expressed in (4.8).

□

4.7 Convergence time and approximation algorithm

In this section we show that under **ex**, starting from any initial assignment, if we let players take the maximum-gain responses, then in polynomially many steps, the outcome is an assignment whose cost under **ex** is at most $1+\phi+\varepsilon$ times the optimal cost. This of course implies a polynomial time algorithm of the same ratio.

Definition 4.7.1. *Given an assignment* $\mathbf{x}$ *and suppose that job* $i \in X_{j'}$*, let*

$$\Delta_i^{\mathbf{ex}}(\mathbf{x}) = w_i C_i^{\mathbf{ex}}(\mathbf{x}) - \tau,$$

where

$$\tau = \max_{j' \in \mathcal{M}} w_i(p_{ij} + \sum_{\substack{k \in X_j,\\ \rho_{kj} < \rho_{ij}}} p_{kj}) + p_{ij} \sum_{\substack{k \in X_j \setminus \mathcal{J}(\alpha(i)),\\ \rho_{kj} > \rho_{ij}}} w_k - p_{ij'} \sum_{\substack{k \in X_{j'}(\alpha(j)),\\ \rho_{kj'} > \rho_{ij'}}} w_k.$$

It can be observed that the definition $\Delta_i^{\mathbf{ex}}(\mathbf{x})$ is the maximum possible decrease in the cost of player $\alpha(i)$ when she considers moving of her job i. Clearly $\Delta_i^{\mathbf{ex}}(\mathbf{x}) \geq 0$. Let $\Delta^{\mathbf{ex}}(\mathbf{x}) = \sum_{i \in \mathcal{J}} \Delta_i^{\mathbf{ex}}(\mathbf{x})$ and we say $\mathbf{x}$ is an ε-WE if $\Delta^{\mathbf{ex}}(\mathbf{x}) < \varepsilon C^{\mathbf{ex}}(\mathbf{x})$.

Theorem 4.7.2. *Let* $\mathbf{x}(0)$ *be any initial assignment and let* $\hat{\mathbf{x}}$ *be the global minimizer of* $\Phi(\cdot)$*, where* Φ *is the potential function of* **ex**. *Then for any* $\varepsilon > 0$*, maximum-gain best response dynamics generates an* ε*-WE* $\mathbf{x}$ *in at most* $O(\frac{n}{\varepsilon} \log \frac{\Phi(\mathbf{x}(0))}{\Phi(\hat{\mathbf{x}})})$ *steps. This* ε*-equilibrium satisfies* $C^{\mathbf{ex}}(\mathbf{x}) \leq (1 + \phi + O(\varepsilon)) C^{\mathbf{sr}}(\mathbf{x}^*)$*.*

Proof. First suppose that $\mathbf{x}$ is an ε-weak-equilibrium. For each job $i \in X_j^*$, suppose that $i \in X_{j'}$, and define

$$\pi(i) = w_i(p_{ij} + \sum_{\substack{k \in X_j,\\ \rho_{kj} < \rho_{ij}}} p_{kj}) + p_{ij} \sum_{\substack{k \in X_j \setminus \mathcal{J}(\alpha(i)),\\ \rho_{kj} > \rho_{ij}}} w_k - p_{ij'} \sum_{\substack{k \in X_{j'}(\alpha(i)),\\ \rho_{kj'} > \rho_{ij'}}} w_k.$$

Observe that the definition of $\pi(i)$ is simply the RHS of the inequality in Lemma 4.5.2. As shown in Lemma 4.5.3 and Theorem 4.5.4[1],

$$\sum_{i \in \mathcal{J}} \pi(i) \leq \frac{1}{4\beta} C^{\mathbf{ex}}(\mathbf{x}) + (1 + \beta) C^{\mathbf{sr}}(\mathbf{x}^*).$$

By the definition of $\Delta^{\mathbf{ex}}(\mathbf{x})$ and the fact that $\mathbf{x}$ is an ε-weak equilibrium,

$$\varepsilon C^{\mathbf{ex}}(\mathbf{x}) > \Delta^{\mathbf{ex}}(\mathbf{x}) \geq C^{\mathbf{ex}}(\mathbf{x}) - \sum_{i \in \mathcal{J}} \pi(i) \geq (1 - \frac{1}{4\beta}) C^{\mathbf{ex}}(\mathbf{x}) - (1 + \beta) C^{\mathbf{sr}}(\mathbf{x}^*).$$

Rearranging terms, we obtain that

$$C^{\mathbf{ex}}(\mathbf{x}) \leq \frac{1 + \beta}{1 - \frac{1}{4\beta} - \varepsilon} C^{\mathbf{sr}}(\mathbf{x}^*) \leq (1 + \phi + O(\varepsilon)) C^{\mathbf{sr}}(\mathbf{x}^*),$$

[1]Observe that the sum $\sum_{i \in \mathcal{J}} \pi(i)$ is just the expression in (4.7) and the rest of the derivation on $\sum_{i \in \mathcal{J}} \pi(i)$ in the proof of Lemma 4.5.3 and Theorem 4.5.4 does not depends on the fact that $\mathbf{x}$ is a WE.

where β is set to be $\frac{1+\sqrt{5}}{4}$. This proves the second part of the theorem.

Next we argue that an ε-WE can be reached in the stated number of steps. For this suppose that $\mathbf{x}(t)$ is the assignment after t steps of the dynamics. Suppose further that $\mathbf{x}(t)$ is not an ε-WE, so that $\Delta^{\mathbf{ex}}(\mathbf{x}(t)) > \varepsilon C^{\mathbf{ex}}(\mathbf{x}(t))$. Consider the job i such that moving job i decreases the cost of a player the most; for this job we have $\Delta_i^{\mathbf{ex}}(\mathbf{x}(t)) > (\varepsilon/n)C^{\mathbf{ex}}(\mathbf{x}(t))$. Now by the facts that $C^{\mathbf{ex}}(\mathbf{x}(t)) \geq \Phi(\mathbf{x}^t)$ and $\Phi(\mathbf{x}(t+1)) = \Phi(\mathbf{x}(t)) - \Delta_i^{\mathbf{ex}}(\mathbf{x}(t)_))$, we derive

$$\Phi(\mathbf{x}(t+1)) \leq (1 - \frac{\varepsilon}{n})\Phi(\mathbf{x}(t)).$$

Thus if no ε-WE is found in the first T steps,

$$\Phi(\hat{\mathbf{x}}) \leq \Phi(\mathbf{x}(T)) \leq (1 - \frac{\varepsilon}{n})^T \Phi(\mathbf{x}(0)).$$

This yields the required bound on the number of steps. □

Theorem 4.7.3. $\frac{\Phi(\mathbf{x}(0))}{\phi(\hat{\mathbf{x}})} \leq \frac{np_{max}}{p_{min}}$, *where* $\mathbf{x}(0)$ *is any initial assignment and* $\hat{\mathbf{x}}$ *is the global minimizer of the potential function,* $p_{max} = \max_{j\in\mathcal{M}, i\in\mathcal{J}} p_{ij}$ *and* $p_{min} = \min_{j\in\mathcal{M}, i\in\mathcal{J}} p_{ij}$

Proof. The potential function can be upper-bounded as follows. All jobs belong to different players and all are assigned to the same machine, and all have the processing times of p_{max} on it. Thus $\Phi(\mathbf{x}(0)) \leq C^{\mathbf{ex}}(\mathbf{x}(0)) \leq (\sum_{i\in\mathcal{J}} w_i)(np_{max})$.

On the other hand, the potential function can be lower-bounded as follows: each job is assigned to a different machine and all have the same processing times p_{min} on them. Thus $\Phi(\hat{\mathbf{x}}) \geq \Gamma(\hat{\mathbf{x}}) \geq (\sum_{i\in\mathcal{J}} w_i)(p_{min})$. The proof follows.

□

4.8 Lower bounds

In this section we prove the lower bounds claimed in the chapter. We start by showing that for WE, the coordination ratio of any prompt mechanisms (randomized or deterministic) is at least $1 + \phi \approx 2.618$, where ϕ is the golden ratio. The construction is an adaptation of a result by Caragiannis et al (12). Then, by twisting the same instance we get a lower bound of 5.828 instance for **ps**. Additionally we show that the coordination ratio of **ps** is higher than 2.618 even under NE, indeed it is at least $2\sqrt{2} \approx 2.824$. Finally, we show that the coordination ratio of any anonymous prompt mechanism is at least 4 for WE.

4.8.1 Prompt mechanisms

Let ϕ be the golden ratio, i.e., the positive solution of $\phi^2-\phi-1=0$, and consider the following instance of the restricted related machines model. We have a path of n nodes with $n-1$ edges. Node m_i, $0 \leq i \leq n-2$ represents a machine with speed ϕ^{2i}. Node m_{n-1} represents a machine with speed $\phi^{2(n-2)}$. On the other hand, edge j_i, $0 \leq i \leq n-2$, is a job that can only go to machine m_i and machine m_{i+1}. The processing time and the weight of job j_i equals ϕ^i. In this instance there is only one player that controls all the jobs.

Suppose that each job j_i is assigned to machine m_i for $0 \leq i \leq n-2$. The cost for job j_i under this assignment equals $w_{j_i}p_{j_i}/\phi^{2i} = 1$, so that such an assignment has total cost of $n-1$. We claim that this is actually a WE. To see this assume the player flips a single job j_i from machine m_i to machine m_{i+1}, for some $0 \leq i \leq n-2$. Consider first flipping a job j_i, for $0 \leq i \leq n-3$, to machine m_{i+1}. Since the mechanism is prompt, before the flip jobs where finishing by their processing times and thus the aggregate cost of jobs j_i and j_{i+1} was 2. After the change, any feasible policy has only two possible choices:

1. Finish job j_i before job j_{i+1}, i.e., $C_{j_i} \leq C_{j_{i+1}}$. Since the mechanisms is feasible in this case we have that $p_{j_i} \leq C_{j_i}$ and $p_{j_i}+p_{j_{i+1}} \leq C_{j_{i+1}}$. Therefore the aggregate cost of these jobs satisfies:
$$w_{j_i}C_{j_i} + w_{j_{i+1}}C_{j_{i+1}} \geq \frac{\phi^{2i} + (\phi^i + \phi^{i+1})\phi^{i+1}}{\phi^{2i+2}} = \phi^2 = 2.$$
2. Finish job j_{i+1} before job j_i, i.e., $C_{j_{i+1}} \leq C_{j_i}$. Since the mechanisms is feasible in this case we have that $p_{j_{i+1}} \leq C_{j_{i+1}}$ and $p_{j_i}+p_{j_{i+1}} \leq C_{j_i}$. Therefore the aggregate cost of these jobs satisfies:
$$w_{j_i}C_{j_i} + w_{j_{i+1}}C_{j_{i+1}} \geq \frac{\phi^i(\phi^i + \phi^{i+1}) + \phi^{2i+2}}{\phi^{2i+2}} = 2.$$

In none of the cases, the total cost of jobs j_i and j_{i+1} decreases after the change, and thus, even if randomization is allowed the situation was a WE. It is also clear that the player also has no incentive to move job j_{n-2} from machine m_{n-2} to m_{n-1}. Therefore, we have a WE.

To conclude, observe that the cost of the optimal assignment, which assigns job j_i to machine m_{i+1}, for all $0 \leq i \leq n-2$, is
$$\sum_{i=0}^{n-3}\left(\frac{\phi^{2i}}{\phi^{2i+2}}\right) + \frac{\phi^{2n-4}}{\phi^{2n-4}} = \frac{n-2}{\phi^2} + 1.$$

Therefore, the coordination ratio is at least $\frac{n-1}{\frac{n-2}{\phi^2}+1} \to \phi^2 = 1 + \phi$.

4.8.2 ps under WE

To obtain a bound on proportional sharing we take the same instance as before but let change the value of ϕ to be $1 + \sqrt{2}$.

Arguing in a similar way as before, assigning j_i to machine m_i for $0 \le i \le n-2$ is a WE, with total cost $n-1$. If not, the player can flip her job j_i from machine m_i to machine m_{i+1}, for some $0 \le i \le n-2$. If we consider j_i, for some $0 \le i \le n-3$, to machine m_{i+1}. The cost of jobs j_i and j_{i+1} changes from 2 to $\frac{((1+\sqrt{2})^i+(1+\sqrt{2})^{i+1})^2}{(1+\sqrt{2})^{2i+2}} = 2$. Also the player also has no incentive to move job j_{n-2} from machine m_{n-2} to m_{n-1}. We conclude by noting that the cost of the optimal assignment, which assigns job j_i to machine m_{i+1} is

$$\sum_{i=0}^{n-3} \left(\frac{(1+\sqrt{2})^{2i}}{(1+\sqrt{2})^{2i+2}} \right) + \frac{(1+\sqrt{2})^{2n-4}}{(1+\sqrt{2})^{2n-4}} = \frac{n-2}{(1+\sqrt{2})^2} + 1.$$

Therefore, the coordination ratio is at least $\frac{n-1}{\frac{n-2}{(1+\sqrt{2})^2}+1} \to (1+\sqrt{2})^2 = 3 + 2\sqrt{2}$.

4.8.3 ps under NE

Consider the following instance of the restricted related machines model. We have a binary tree of d levels. Let the root be in level 0 and the leaves be in level $d-1$. Every nodes in level $l, 0 \le l \le d-2$ represents a machine with speed S^l where $S = \frac{2+\sqrt{2}}{2}$. The nodes in levels $d-1$ represent machines with speed $L \cdot S^{d-2}$ where $L = \frac{3\sqrt{2}-2}{2}$. Every edge in the graph represents a job of size and weight 1 that can only go to the two machines on its endpoints. We call these jobs the big jobs. On every machine we also have $\sqrt{2}/\varepsilon$ jobs that can only go to this machine each of size and weight ε. We call these jobs the small jobs. Finally, for each machine there is a player controlling all small jobs in that machine and the single big job going to its parent node. The player in the root only controls small jobs.

Suppose that each big job is assigned to the machine in the parent node. Such as assignment has cost of

$$\begin{aligned}
&\sum_{i=0}^{d-2} \left(2^i \frac{(2+\sqrt{2})^2}{S^i} \right) + 2^{d-1} \frac{2}{L \cdot S^{d-2}} \\
&= \left(\frac{2}{S}\right)^{d-1} (3+2\sqrt{2}) \left((2+\sqrt{2})^2 + \frac{2}{1+2\sqrt{2}} \right) - 2(3+2\sqrt{2})^2.
\end{aligned}$$

We claim that this is a NE. Indeed, if a player on a machine in level $1 \leq i \leq d-2$ flips her big job from the parent machine. The cost of her jobs goes from $\frac{2+\sqrt{2}}{S^{i-1}} + \frac{\sqrt{2}(2+\sqrt{2})}{S^i}$ to $\frac{(1+\sqrt{2})(3+\sqrt{2})}{S^i}$. So the total cost of this player is not improved after the change. Now consider the case where the player is on a machine in level $d-1$. Before the change, the total cost of the player was $\frac{2+\sqrt{2}}{S^{d-2}} + \frac{2}{L \cdot S^{d-2}}$, while after the change the cost becomes $\frac{(1+\sqrt{2})(1+\sqrt{2})}{L \cdot S^{d-2}}$. Again the total cost of the player is not improved after the change. Therefore, we have a NE.

Finally, observe that the cost of the optimal assignment, which assigns each big job to the machine in the child node, is

$$\begin{aligned}
&1 + \sum_{i=1}^{d-2} \left(2^i \frac{1+\sqrt{2}(1+\frac{\sqrt{2}}{2})}{S^i} \right) + 2^{d-1} \frac{1+\sqrt{2}(1+\frac{\sqrt{2}}{2})}{L \cdot S^{d-2}} + O(\varepsilon) \\
&= \left(\frac{2}{S}\right)^{d-1} (3+2\sqrt{2}) \left(2+\sqrt{2}+\frac{2+\sqrt{2}}{1+2\sqrt{2}}\right) - 11 - 8\sqrt{2} + O(\varepsilon).
\end{aligned}$$

Letting $\varepsilon \to 0$, the additive $O(\varepsilon)$ term can be ignored. A straightforward calculation amounts to conclude that as $d \to \infty$, the coordination ratio approaches $2\sqrt{2} \approx 2.828$.

4.8.4 Anonymous mechanisms

Consider the following instance of the restricted related machines model. We have a binary tree of d levels, with an additional layer of 2^{d-1} nodes, each connected with one edge to a leaf of the binary tree. So we have $d+1$ levels and $2^d - 1 + 2^{d-1}$ nodes. For the original binary tree let the root be in level 0 and the leaves be in level $d-1$. Every nodes in level $l, 0 \leq l \leq d-2$ represents a machine with speed 2^l. The nodes in levels $d-1$ and d represent machines with speed $\frac{3}{2} \cdot 2^{d-2}$. As usual, every edge in the graph represents an unweighted job of size 1 that can only go to the two machines on its endpoints. In this situation we have 2^{d-1} players. Every player controls a set of jobs that lies on a path between a non-leaf node and a leaf node of the graph. This is done as follows: the first player owns all jobs is a path from the root to a leaf; then remove these jobs (edges) and consider a path in the remainder from a node as close as possible to the root to a leaf and assign them to a player; the procedure continues until no jobs are left. Figure 4.1 depicts the instance when $d = 4$. The numbers inside the nodes represent the speed of the machines and the letters on the edges represent the player controlling the job.

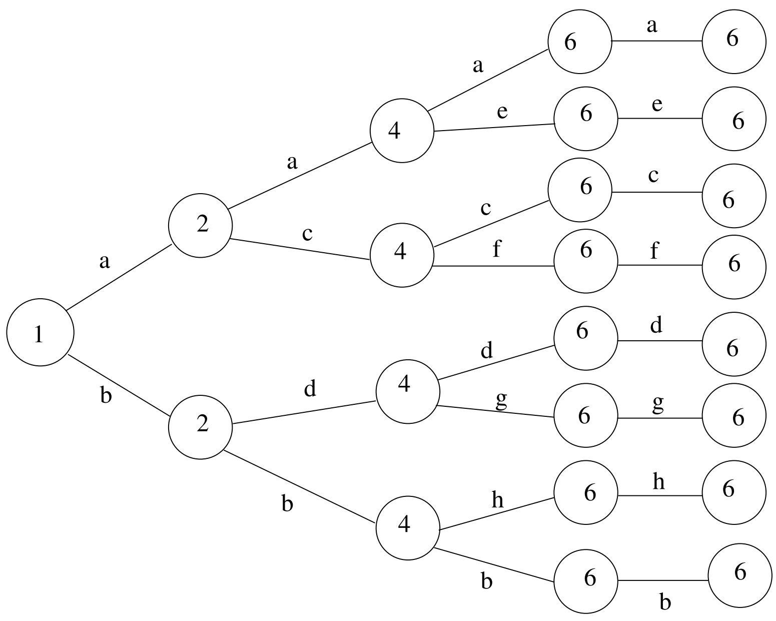

Figure 4.1: The lower bound instance with $d = 4$.

We claim that if each job is assigned to the machine in the level with a smaller index, i.e., closer to the root, we have a WE, whose total cost is:

$$\sum_{i=0}^{d-2}\left(\frac{2\cdot 2\cdot 2^i}{2^i}\right)+\frac{2^{d-1}}{\frac{3}{2}2^{d-2}}=4(d-1)+\frac{4}{3}.$$

If this is not a WE a player can flip one job from a machine in level i to a machine in level $i+1$, for some $0 \leq i \leq d-1$. Consider the case where $0 \leq i \leq d-3$. Before the change, the aggregate cost of jobs in levels i and $i+1$ was $\frac{2}{2^i}+\frac{2}{2^{i+1}}$. After the change the cost is $\frac{2\cdot 3}{2^{i+1}}$. So the total cost of the player does not decrease. Now consider the case where the player can flip the job from level $d-2$ to level $d-1$. Before the change, the cost of jobs in level $d-2$ and level $d-1$ is $\frac{2}{2^{d-2}}+\frac{1}{(3/2)2^{d-2}}$. After the change the cost is $\frac{4}{(3/2)2^{d-2}}$. So again the total cost of the player is not improved after the change . Finally, it is easily verified that the player has no incentive to move the job from machine in level $d-1$ to d. Therefore, we have a WE.

We conclude, observing that the cost of the optimal assignment, which assigns each job to the machine in the level with a larger index, i.e., farther from the root, is

$$\sum_{i=1}^{d-2}\left(\frac{2^i}{2^i}\right)+2\frac{2^{d-1}}{\frac{3}{2}2^{d-2}}=d-2+\frac{8}{3}.$$

Therefore, the coordination ratio is at least $\frac{4(d-1)+\frac{4}{3}}{d-2+\frac{8}{3}}$ which approaches 4 for large d.

4.9 Final remarks

We have proved that **sr** is the best possible non-preemptive policy, and to beat its coordination ratio we have used **ex**, a policy that, as opposed to **sr**, importantly relies on who owns which job. We conjecture that if we restrict to policies that ignore the ownership of the jobs the ratio of 4 cannot be improved. This is indeed the case for non-preemptive policies, and also for fully preemptive policies as shown in Section 4.8. Also, for natural policies with this property such as **ps** or the RAND policy (18) the technique in this chapter only lead to larger bounds.

Our lower bound on general prompt seems to be the natural limit. Non prompt policies that are allowed to use global information can certainly beat this as they can simply introduce very large delays for jobs that are not assigned to it in an optimal schedule. By doing this, such policies can easily achieve low coordination ratio (say optimal if they have unlimited computational power or 3/2 if they use the best known approximation algorithms. It would be interesting to explore what happens with this non prompt policies when they can only use local information.

Another interesting question refers to the quality of the actual NE of this game. Of course our upper bounds applies to that equilibrium concept, but is may be possible that the coordination ratio of **ex** for the NE is below 2.618. We know however that this cannot be better than 2.5 as we show in the chapter.

Finally, we note that by mimicking the analysis in (18) we obtain a similar $2+\varepsilon$ approximation algorithm for $R| \,|\sum w_i C_i$, independent of which jobs belong to which players. It is possible that by carefully choosing the game structure this can be beaten.

Bibliography

[1] F. Abed, J. Correa, and C.-C. Huang. Optimal coordination mechanisms for multi-job scheduling games. In *22nd Annual European Symposium on Algorithms (ESA)*, 2014. 6

[2] F. Abed, C.-C. Huang. Preemptive coordination mechanisms for unrelated machines. In *20th Annual European Symposium on Algorithms (ESA)*, 2012. 5, 24

[3] F. Abed, C.-C. Huang. Coordinating oligopolistic players in unrelated machine scheduling. Theor. Comput. Sci. 570: 40-54 (2015) 5

[4] E. Anshelevich, B. Caskurlu, and A. Hate. Partition equilibrium always exists in resource selection games. In *SAGT*, pages 42–53, 2010. 26

[5] B. Awerbuch, Y. Azar, Y. Richter, and D. Tsur. Tradeoffs in worst-case equilibria. *Theor. Comput. Sci.*, 361(2-3):200–209, 2006.

[6] Y. Azar, K. Jain, and V.S. Mirrokni. (Almost) Optimal coordination mechanisms for unrelated machine scheduling. In *SODA* 2008. 4, 7, 8, 9, 12, 20, 21, 22, 24

[7] U. Bhaskar, L. Fleischer, D. Hoy, and C.-C. Huang. Equilibria of atomic flow games are not unique. In *SODA*, pages 748–757, 2009. 26

[8] S. Bhattacharyay, S. Imz, J. Kulkarnix, K. Munagala. Coordination mechanisms from (almost) all scheduling policies. In *ITCS* 2014. 4

[9] T. Brueggemann, J.L. Hurink, W. Kern. Quality of move-optimal schedules for minimizing total weighted completion time. *Oper. Res. Lett.*, 34(5):583-590, 2006. 4, 49

[10] J. Bruno, E.G. Coffman, and R. Sethi. Scheduling independent tasks to reduce mean finishing time. *Commun. ACM*, 17:382–387, 1974. 1

[11] I. Caragiannis. Efficient coordination mechanisms for unrelated machine scheduling. In *SODA* 2009. 4, 8, 9, 11, 24, 26, 27, 28, 40, 42

[12] I. Caragiannis, M. Flammini, C. Kaklamanis, P. Kanellopoulos, and L. Moscardelli. Tight bounds for selfish and greedy load balancing. *Algorithmica*, 61(3):606–637, 2011. 4, 60

[13] B. Chen, C.N. Potts, G.J. Woeginger. A review of machine scheduling: Complexity, algorithms and approximability. In Handbook of Combinatorial Optimization (volume 3), Kluwer Academic Publishers. 1998. 4, 49

[14] G. Christodoulou, E. Koutsoupias, A. Nanavati. Coordination mechanisms. *Theor. Comput. Sci.*, 410(36):3327–3336, 2009. Preliminary version in *ICALP* 2004. 4, 7

[15] J. Cohen, C. Dürr, and N.K. Thang. Non-clairvoyant scheduling games. *Theory Comput. Syst.*, 49(1):3–23, 2011. 8, 24, 25

[16] J. Cohen, C. Dürr, N.K. Thang. Smooth inequalities and equilibrium inefficiency in scheduling games. In *WINE* 2012. 4, 24

[17] R. Cole, J.R. Correa, V. Gkatzelis, V.S. Mirrokni, N. Olver Inner product spaces for MinSum coordination mechanisms. In *STOC* 2011. 3, 4, 13, 24

[18] R. Cole, J.R. Correa, V. Gkatzelis, V. Mirrokni, N. Olver. Decentralized utilitarian mechanisms for scheduling games. *Game. Econ. Behav.*, to appear. 3, 4, 48, 49, 51, 54, 65

[19] R. Cominetti, J. Correa, and N. Stier-Moses. The impact of oligopolistic competition in networks. *Operations Research*, 57(6):1421–1437, 2009. 26

[20] J. Correa, M. Queyranne. Efficiency of equilibria in restricted uniform machine scheduling with total weighted completion time as social cost. *Naval Res. Logist.*, 59:384–395, 2012. 4, 48

[21] A. Czumaj, B. Vöcking. Tight bounds for worst-case equilibria. *ACM T. Algo.*, 3, 2007. 4

[22] E. Davis, J.M. Jaffe. Algorithms for scheduling tasks on unrelated processors. *J. ACM*, 28(4):721–736, 1981. 1

[23] C. Dürr, N.K. Thang. Non-clairvoyant scheduling games. In *SAGT* 2009. 4, 49, 54

[24] B. Farzad, N. Olver, A. Vetta. A priority-based model of routing. *Chic. J. Theor. Comput.*, 2008. 4

[25] M. Feldman and M. Tennenholtz. Partition equilibrium. In *SAGT*, pages 48–59, 2009. 26

[26] G. Finn, E. Horowitz. A linear time approximation algorithm for multiprocessor scheduling. *BIT*, 19:312–320, 1979. 1

[27] L. Fleischer, Z. Svitkina. Preference-constrained oriented matching. In *ANALCO* 2010. 4, 8, 9, 12, 24

[28] D. Fotakis, S. Kontogiannis, and P. Spirakis. Atomic congestion games among coalitions. *ACM Transactions on Algorithms*, 4(4), 2008. The conference version appeared in ICALP 2006. 26

[29] M. Garey and D. Johnson. *Computers and Intractablility.* Freeman, 1979. 42

[30] A. Hayrapetyan, É. Tardos, and T. Wexler. The effect of collusion in congestion games. In *STOC*, pages 89–98, New York, NY, USA, 2006. ACM Press. 26

[31] R. Hoeksma, M. Uetz. The Price of Anarchy for Minsum related machine scheduling. In *WAOA* 2011. 4

[32] H. Hoogeveen, P. Schuurman, G.J. Woeginger. Non-approximability results for scheduling problems with minsum criteria. In *IPCO* 1998. 1

[33] W.A. Horn. Minimizing average flow time with parallel machines. *Oper. Res.*, 21(3):846–847, 1973. 1

[34] O.H. Ibarra, C.E. Kim. Heuristic algorithms for scheduling independent tasks on nonidentical processors. *J. ACM*, 24(2):280–289, 1977. 1

[35] N. Immorlica, L. Li, V.S. Mirrokni, and A.S. Schulz. Coordination mechanisms for selfish scheduling. *Theor. Comput. Sci.*, 410(17):1589–1598, 2009. 4, 8, 11, 13, 24

[36] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *STACS* 1999. 2, 3, 8

[37] J. Lenstra, D. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46:259–271, 1990. 1, 7

[38] P. Lu, C. Yu. Worst-case Nash equilibria in restricted routing. In *WINE*, 231–238, 2008. 4

[39] J. Nash. Equilibrium points in N-person games. *PNAS*, 36, 48-49, 1950. 47

[40] J. Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, 1951.

[41] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007. 4

[42] C.N. Potts, V. Strusevich. Fifty years of scheduling: a survey of milestones. *J Oper. Res. Society*, 60(1):41-68, 2009. 4, 49

[43] M. Rahn, G. Schäfer. Bounding the inefficiency of altruism through social contribution games. Manuscript 2013. 4

[44] T. Roughgarden. *Selfish Routing and the Price of Anarchy*. The MIT Press, 2005. 26

[45] T. Roughgarden. Intrinsic robustness of the price of anarchy. In *STOC* 2009. 48

[46] S. Sahni, Y. Cho. Bounds for list schedules on uniform processors. *SIAM J. Comput.*, 9:91–103, 1980. 1

[47] A.S. Schulz, M. Skutella. Scheduling unrelated machines by randomized rounding. *SIAM J. Discrete Math.*, 15(4):450–469, 2002. 2

[48] P. Schuurman, T. Vredeveld. Performance guarantees of local search for multiprocessor scheduling. In *IPCO* 2001. 2

[49] J. Sethuraman, M.S. Squillante. Optimal scheduling of multiclass parallel machines. In *SODA* 1999. 2

[50] M. Skutella. Convex quadratic and semidefinite programming relaxations in scheduling. *J. ACM*, 48(2):206–242, 2001. 2

[51] M. Skutella, G.J. Woeginger. A ptas for minimizing the total weighted completion time on identical parallel machines. *Math. Oper. Res.*, 25(1):63–75, 2000. 2

[52] W. Smith. Various optimizers for single stage production. *Naval Res. Logist. Quart.*, 3(1-2):59–66, 1956. 1

[53] T. Vredeveld, C. Hurkens. Experimental comparison of approximation algorithms for scheduling unrelated parallel machines. *INFORMS J. Comput.*, 14:175–189, 2002. 48, 49

[54] Wikipedia. http://en.wikipedia.org/wiki/Independence of irrelevant alternatives. 11