Master Thesis

Price of Anarchy for Machine Scheduling Games with Sum of Completion Times Objective

Ruben Hoeksma

Graduation Committee:
prof.dr. M.J. Uetz
prof.dr. J.L. Hurink
dr. B. Manthey

October 26, 2010

UNIVERSITY OF TWENTE.

Voor Nan

Preface

This master thesis is the final result of my research on the price of anarchy for uniformly related machine scheduling with sum of completion times objective. The research was done at the chair of Discrete Mathematics and Mathematical Programming at the University of Twente. I would like to use this opportunity to thank the people who supported me during the research for and the writing of this thesis.

First of all I want to thank Marc Uetz for introducing me to the realm of algorithmic game theory and, especially, for coming up with the research topic of this thesis. Also many thanks for the constructive criticism and useful deliberations.

I would also like to thank the rest of the staff and the master and Ph.D. students from the chair of Discrete Mathematics and Mathematical Programming for a good working environment and interesting discussions when we were not working.

Finally, I am very grateful to my family and my friends who supported me during my whole study and, especially, while I was working on this thesis.

Ruben Hoeksma
Enschede, October 2010

Abstract

The *uniformly related machine scheduling* model with sum of completion times objective is well known and its optimal solution is easy to find. However, this solution requires a central administrator that schedules the jobs on the machines. We look at this model from a game theoretical point of view and introduce for each job a selfish agent, which chooses the machine which processes the job, and is only interested in minimizing the completion time of its own job. Our interest lies in the *price of anarchy* for this game. That is, the ratio between the objective value of the optimal solution and the objective value of the Nash equilibrium of the game.

Recent work has shown that for the more general unrelated machine scheduling model, with the same objective, the price of anarchy is at least 3 and at most 4. For the uniformly related machine scheduling game, with only two machines we prove an upper bound on the price of anarchy of $\frac{1}{2} + \frac{1}{2}\sqrt{5} \approx 1.6180$. We also construct an instance which has price of anarchy equal to $\frac{3}{2} \cdot \frac{2+\sqrt{3}}{3+\sqrt{3}} \approx 1.1830$. For the general case with any number of machines we show an upper bound of 3. Furthermore, we construct a set of instances, which price of anarchy can be made arbitrarily close to $\frac{e}{e-1} \approx 1.5820$.

Contents

1 Introduction **1**

1.1 Machine scheduling . . . 2

1.2 Scheduling games . . . 4

1.3 Outline of the thesis . . . 6

2 The problem setting **7**

2.1 Uniformly related machine scheduling . . . 7

2.2 Single machine model . . . 8

2.3 Optimal solution algorithm . . . 9

2.4 The scheduling game . . . 10

2.5 Equilibrium solution algorithm . . . 12

2.6 Robust price of anarchy . . . 13

2.7 Related Work . . . 14

3 Instances with large price of anarchy **17**

3.1 Reducing the set of instances . . . 17

3.2 Simple instances . . . 19

3.3 Instances with all jobs on the fastest machine in equilibrium . . . 23

4 Upper bounds on the price of anarchy **29**

4.1 Known bounds . . . 29

4.2 Unit speeds and unit lengths . . . 29

4.3 Upper bounds on the price of anarchy for the $Q||\sum C_j$ game . . . 33

5 Concluding remarks **43**

A Used notation **45**

B Proofs of analysis steps **47**

C $Q||\sum C_j$ is $(2, \frac{1}{2})$-smooth **49**

Bibliography **52**

Chapter 1

Introduction

In many classical applications of optimization problems it suffices to know the choices that should be made to get a desirable outcome. When these choices are known a single administrator of the system simply implements these choices and the desirable outcome follows. But, with recent developments in computer science, especially with respect to the Internet, many situations arise in which not a single administrator makes all choices, but many individual, selfishly acting, agents make their own choices. The following fictitious example illustrates a problem in which such a single administrator may not exist.

Example 1.1. *Consider a university with a number of copy shops. Every copy shop has a printer which can print any print job, but some of the machines are old and therefore less fast. Several students want to print their report in one of the shops. If they supply their files today their work will be printed the next day. Some reports are very long and cumbersome, some are brief and to the point. Every student wants his or her report printed as soon as possible.*

The students may ask one representative to schedule all the print jobs to the copy shops in such a way that the total waiting time is minimal. Or they may choose to, each on their own, send their files to one of the copy shops to have them printed.

While in this example the introduction of a single administrator would be fairly easy, in general this may not be the case. With this in mind, an interesting question is if the lack of such an administrator will lead to worse solutions. Especially, we would like to be able to quantify how much the solution would possibly get worse, when a such a single administrator is missing. A measure used for this is the *price of anarchy.* In this thesis we analyze the price of anarchy for the, fairly simple, *uniformly related machine schedule* game, with sum of completion times as objective function.

In this first chapter, we give a short introduction into general machine scheduling, scheduling games and the principle of the price of anarchy. In section 1.3 we give a short outline of the rest of the thesis.

1.1 Machine scheduling

The students of example 1.1 want to minimize their total waiting time. When they ask one representative to try and achieve this, the problem can be modeled by the machine scheduling model known as *uniformly related machine scheduling* model.

The uniformly related machine scheduling model is a classical machine scheduling model. In general a machine scheduling model treats n jobs that have to be scheduled on m machines. For every combination of a job j and a machine i, p_{ij} is the time it takes to process job j on machine i. In other words p_{ij} is the jobs processing time on machine i.

Throughout the years many different scheduling models have been studied. Most of these models have some properties in common, which makes a general notation useful. The most commonly used notation is that of Graham, Lawler, Lenstra and Rinnooy Kan, described in [GLLRK79]. The notation characterizes a machine scheduling model by three parts. Namely, the machine environment, the job properties and the objective of the schedule. For each of these parts a separate field is used. The notation is of the form $\alpha|\beta|\gamma$. The α-field for the machine environment, the β-field for properties of the jobs and the γ-field for the objective function.

To give some notion of the many different possible scheduling models, some of the possibilities for each of the fields are given below.

1.1.1 α-field

The α-field contains the machine environment. It specifies the machines on which the jobs have to be scheduled. There may be just a single machine, denoted by a 1, or there may be more. The identical parallel machine model has multiple machines, each with equal speed. Identical parallel machines are denoted by a P. A parallel machine model with a specific number of machines m is denoted by Pm.

A first expansion of the identical parallel machine model leads to uniformly related machines[1]. It is denoted by a Q in the α-field. The uniformly related machine model has multiple machines, each with its own speed s_i. With each job j having a fixed length p_j, the processing time of job j on machine i then is $p_{ij} = p_j/s_i$.

The unrelated machine model is denoted by an R in the first field. It may be seen as an expansion of the uniformly related machine model. Within the unrelated machine model no constraints exist on the p_{ij}'s. For any pair i, j, p_{ij} may have any value. A consequence of this is, that, a general order of the jobs on processing time is not possible.

Of course many other machine configurations are possible. For instance, all of the above considered machine configurations are single stage problems, whereas

[1]Exact naming of the models varies a lot per publication. Throughout this thesis, we use the naming from this section.

also multi stage models exist. However for this thesis the single, parallel, uniformly related and unrelated machine models are the most relevant. They relate to each other as

$$1 \subseteq P \subseteq Q \subseteq R.$$

1.1.2 β-field

Properties of jobs are very diverse in nature and many combinations of properties are possible. Some of the most common possibilities are treated here.

If the β-field is empty, the jobs have to be processed as a whole, one at a time per machine. Other possible properties of the jobs are:

$p_{ij} = 1$, $\underline{p} \leq p_{ij} \leq \overline{p}$ The processing times of the jobs are limited to unit processing times ($p_{ij} = 1$) or by an upper and lower bound ($\underline{p} \leq p_{ij} \leq \overline{p}$).

pmtn The jobs can be preempted, meaning that an unfinished job can be set aside, to process another job, and finished later from the point it was preempted.

prec, *tree* The order in which the jobs have to be processed is restricted. The restrictions specify orders between pairs of jobs and can be of arbitrary form (*prec*), or of a specified form, like a tree (*tree*).

r_j Jobs have release dates before which the job cannot be processed.

d_j Jobs have due dates, which specify the time the job should be finished.

Most of the models we study in this thesis have no specific job properties.

1.1.3 γ-field

In general an objective function for a scheduling problem quantifies, in some way, the time it takes the schedule to finish the jobs. The lower the value of the objective function or *objective value*, the better the schedule is. For each goal, different measures are best suitable. Looking at one job j, an obvious measure is the completion time (C_j). Also the lateness and tardiness of j, defined as $L_j = C_j - d_j$ and $T_j = \max(0, C_j - d_j)$, are obvious choices when dealing with jobs with due dates. A last measure is the unit penalty, where

$$U_j = \begin{cases} 1 & \text{if } C_j \leq d_j, \\ 0 & \text{otherwise.} \end{cases}$$

Common objective functions are maximum completion time (*makespan*), $C_{\max}$, and maximum lateness, $L_{\max}$, as well as the sum of completion times ($\sum C_j$), the sum of tardiness ($\sum T_j$) and the summed unit penalty ($\sum U_j$). Sometimes weights are used to express that some jobs have higher priority than others. In that case weighted versions of any objective may be used. For example, weighted sum of completion times is denoted as $\sum w_j C_j$. In this thesis our attention goes to scheduling models with either weighted of non-weighted sum of completion times objective.

1.2 Scheduling games

When the students of example 1.1, instead of letting one person decide the whole schedule, each choose their own copy shop to print their report, they get into a mutual struggle for the best completion time. The traditional scheduling problem now becomes a game between the students. A game in which every student tries to optimize his own goal.

In a scheduling game each job has its own agent, which chooses the machine which will process the job. The choice of an agent is referred to as its strategy. A combination of strategies played by all players is called a strategy profile. Such a strategy profile determines the completion times for each of the jobs. The goal of each agent is minimizing the completion time of the job it represents. The machines on which the jobs are scheduled all use some local rule, which determine the order in which the jobs are processed on that machine. It is common for each machine to use the same rule, but this does not have to be the case. Note that for some models this need not imply that the order of the jobs is the same on each machine.

1.2.1 Nash equilibrium

The *Nash equilibrium* is a widely accepted solution concept for competitive games. It was first suggested by John Nash in [Nas50] and later named after him. The Nash equilibrium concept assumes complete selfishness and no cooperation between the players. A Nash equilibrium is defined as a solution in which no player can improve his utility by changing only his own strategy.

Definition 1 (Nash equilibrium)**.** Let S_j be the set of all possible strategies of player j and let $S = S_1 \times S_2 \times \ldots \times S_n$ be the set of all possible strategy profiles. Then a strategy profile $\nu = (\nu_j, \nu_{-j})$, where ν_j is the strategy of player j and ν_{-j} the strategy profile of all players except j, is a Nash equilibrium if

$$u_j(\nu_j, \nu_{-j}) \geq u_j(\nu'_j, \nu_{-j}) \qquad \forall \nu'_j \in S_j, \forall j \in J, \tag{1.1}$$

where $u_j(\nu)$ is the utility function of player j. Also, ν_j is called a *best response strategy* for j.

We illustrate the concept of a Nash equilibrium with the following example.

Example 1.2 (Prisoners dilemma)**.** *Consider two suspects of a crime, who both face jail time. Both have an equal amount of small crimes, which they certainly will be convicted for. Also, they have participated together in a larger crime for which the public prosecutor will not get a conviction unless one of the suspects tells on the other. So the public prosecutor tells them both that if one tells on his mate he will get two years less jail time on the smaller crimes.*

Now for the smaller crimes the suspects will get 3 years of jail and, for the larger crime they can get 5 years of jail. So if no one tells on the other, both will be in jail for 3 years. However if one tells on the other, and does not get told on *himself, he gets 1 year of jail and the other gets 8 years of jail. Finally, when they both tell on each other, they will both spend the next 6 years in prison.*

	2 tells on 1	2 does not tell on 1
1 tells on 2	(6 years, 6 years)	(1 year, 8 years)
1 does not tell on 2	(8 years, 1 year)	(3 years, 3 years)

Table 1.1: Prisoners dilemma: You should always betray your mate.

Table 1.1 shows in overview the results of the four combinations of choices that the prisoners, in example 1.2, have. It is easy to see that both prisoners will get the lowest sentence, if they tell on the other person, whether or not they themselves get told on. We see that, in the situation where both tell on each other, neither of the suspects can improve his outcome by not telling on the other person. So this situation is in fact a Nash equilibrium. However it is easy to see that both prisoners are better off when they keep their mouths shut and go to jail for 3 years. This notion raises the question of how bad a Nash equilibrium can be, compared to the optimal solution. To measure this we use the *price of anarchy.*

1.2.2 Price of anarchy

While in regular games the only interest of an individual is his own objective, we would like to compare the outcome of a scheduling game to the optimal solution of the corresponding machine scheduling problem. For this we look at a social objective of the game, which here corresponds to the objective function of the scheduling problem. Then the value of this social objective in an equilibrium solution can be compared to the optimal value of the scheduling problem. In [Pap01], Papadimitriou proposed the *price of anarchy* as a measure for this comparison. The price of anarchy is the ratio between the objective values in the equilibrium solution with the highest social objective value and the optimal solution to the scheduling problem.

Definition 2 (Price of Anarchy)**.** The price of anarchy (POA) of a game, G, is the maximal ratio between the equilibrium solution with the highest objective value (NE(I)) and the optimal solution (OPT(I)) of an instance from G.

$$\mathrm{POA}(G) = \max_{I \in G} \frac{\mathrm{NE}(I)}{\mathrm{OPT}(I)} \tag{1.2}$$

We further on refer to the Nash equilibrium that attains (1.2) as the (worst) equilibrium solution.

To illustrate the concept of the price of anarchy we look at Example 1.2. We have already seen that the two suspects will both get a 6 year sentence, when they pursue their own objective. If the two could reach an understanding and do not tell on each other they would both get a 3 year sentence. So, if the social objective is the total jail time they serve together, the price of anarchy of this game is 2.

The price of anarchy has been studied for several types of scheduling games. Most of which were models concerning the makespan objective, see for example [ILMS09]. For the sum of completion times objective the price of anarchy has been studied far less. Furthermore the sum of completion times objective is the so called utilitarian objective, the function that maximizes the total welfare of all players.

1.3 Outline of the thesis

The remainder of this thesis consists of four chapters. In Chapter 2 we treat the exact problem description. We define the machine scheduling model and the scheduling game based on it. We also give the two algorithms that obtain the optimal and the equilibrium solution, together with proofs for their correctness. In the last two sections of Chapter 2 we treat some related work that has already been done.

The results are divided over chapters 3 and 4. In Chapter 3 we start by providing properties of instances with high price of anarchy. Then we show several sets of instances for which we compute the price of anarchy, concluding with the first main result, a set of instances which establish a lower bound on the price of anarchy of $\frac{e}{e-1} \approx 1.5820$. In Chapter 4 then, we start by giving proofs that the price of anarchy for both $P||\sum C_j$ and $Q|p_j = 1|\sum C_j$ is equal to 1. After this we give several different approaches on how to find upper bounds for the price of anarchy and we prove an upper bound of 3 for the general $Q||\sum C_j$ model.

Chapter 5 summarizes the results and gives some remarks on possible future research.

Chapter 2

The problem setting

2.1 Uniformly related machine scheduling

The subject of this thesis is the uniformly related machine scheduling model with sum of completion times objective. In the notation of Graham et al. ([GLLRK79]) the model is denoted by $Q||\sum C_j$.

The model considers n jobs, which have to be processed on m machines. Each job j has a length, p_j, representing the amount of work that has to be done to finish the job. The jobs are indexed, $1, \ldots, n$, such that $p_1 \leq \ldots \leq p_n$. The set of all jobs is denoted by J. Each machine i can process a certain amount of work per unit of time, represented by its speed, s_i. The machines are indexed, $1, \ldots, m$, such that $s_1 \leq \ldots \leq s_m$. The set of all machines is denoted by M.

The processing time of a job j on a machine i is the length of the job divided by the speed of the machine: $\frac{p_j}{s_i}$.

All jobs have to be processed. Per machine only one job can be processed simultaneously. The objective pursued while scheduling the jobs is minimizing the sum of the completion times of the jobs.

Summarizing:

- Job set: $|J| = n$
- Job lengths: p_j, $j \in J$, $p_1 \leq p_2 \leq \ldots \leq p_n$
- Machine set: $|M| = m$
- Machine speeds: s_i, $i \in M$, $s_1 \leq s_2 \leq \ldots \leq s_m$
- Objective function:

$$\sum_{j \in J} C_j$$

2.2 Single machine model

The single machine case with sum of completion times objective, also denoted as $1||\sum C_j$, treats just one machine on which n jobs have to be scheduled. A job j needs p_j time to be processed on the machine and the objective is still to schedule the jobs in such a way that the sum of completion times is minimized.

A schedule on one machine is just an ordering of the jobs. Let φ be such an ordering. A schedule that follows from the ordering φ might look like the one in Figure 2.1.

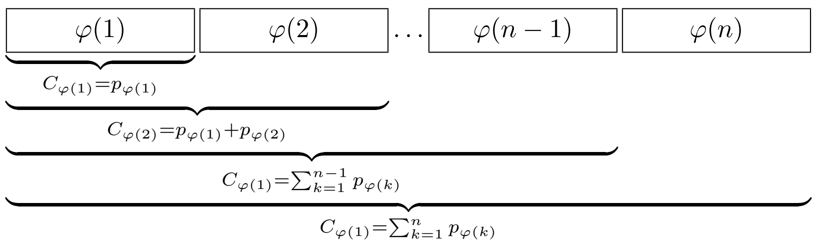
$\varphi(1)$
$\varphi(2)$
$\dots$
$\varphi(n-1)$
$\varphi(n)$
$C_{\varphi(1)}=p_{\varphi(1)}$
$C_{\varphi(2)}=p_{\varphi(1)}+p_{\varphi(2)}$
$C_{\varphi(1)}=\sum_{k=1}^{n-1} p_{\varphi(k)}$
$C_{\varphi(1)}=\sum_{k=1}^{n} p_{\varphi(k)}$

Figure 2.1: A single machine schedule

Figure 2.1 also shows the completion time of each job in the schedule. It is easy to see that the contribution of a job can be measured by its place in the schedule and its processing time. This directly follows from rewriting the objective function

$$\begin{aligned}
\sum_{k=1}^{n} C_{\varphi(k)} &= \sum_{k=1}^{n}\sum_{l=1}^{k} p_{\varphi(l)} \\
&= (p_{\varphi(1)}) + (p_{\varphi(1)} + p_{\varphi(2)}) + \dots + (p_{\varphi(1)} + \dots + p_{\varphi(n)}) \\
&= \underbrace{p_{\varphi(1)} + \dots + p_{\varphi(1)}}_{n \text{ times}} + \underbrace{\varphi(2) + \dots + p_{\varphi(2)}}_{n-1 \text{ times}} + \dots + \underbrace{\varphi(n)}_{1 \text{ time}} \\
&= np_{\varphi(1)} + (n-1)p_{\varphi(2)} + \dots + p_{\varphi(n)} \\
&= \sum_{k=1}^{n}(n-k+1)p_{\varphi(k)}.
\end{aligned}$$

Based on this notion it is easy to prove the optimality of the following algorithm, which is a classical result by Smith [Smi56]. The *Shortest Processing Time* rule, also known as Smith's rule, schedules jobs in order of increasing job length. For later use, the definition below is a generalization on Smith's version to more than one machine.

Algorithm 1 (Shortest Processing Time First Algorithm (SPT))**.** For a uniformly related machine scheduling problem define SPT to be as follows:

1. Take, from the jobs that have not been scheduled yet, the shortest job

2. Schedule this job next on the machine which minimizes its completion time
3. Repeat steps 1 and 2 until all jobs have been scheduled

Theorem 2.1. *For a single machine scheduling problem the SPT algorithm yields an optimal solution*

Proof. Consider an optimal schedule μ. Suppose it is not in SPT order. Then there exist two jobs j and k, such that k is scheduled later than j and $p_k < p_j$. Let q and r denote the number of jobs scheduled after j and k respectively. Then the combined contribution of j and k to the sum of completion times is $qp_j + rp_k$. If the jobs are interchanged within μ the number jobs after any job other than j or k does not change. Interchanging j and k produces a schedule in which the combined contribution of j and k is $rp_j + qp_k$, which is less than $qp_j + rp_k$, since $r < q$ and $p_k < p_j$. So the schedule resulting from the interchange has lower sum of completion times. Therefore μ is not optimal and thus any optimal schedule is an SPT schedule. □

2.3 Optimal solution algorithm

The optimal solution to a general $Q||\sum C_j$ instance can be found using the Minimum Mean Flow Time Algorithm. The description below is based on the one Horowitz and Sahni gave in [HS76].

Algorithm 2 (Minimum Mean Flow Time Algorithm (MFT))**.** For a uniformly related machine scheduling problem define MFT to be as follows:

1. Give every machine a value label $v_i = \frac{1}{s_i}$
2. Take, from the jobs that have not been placed yet, the longest job
3. Add this job to the jobset of the machine with the lowest value label v_i
4. Update the label for that machine to $v_i = v_i + \frac{1}{s_i}$
5. Repeat steps 2 to 4 untill all jobs have been placed
6. For each machine do SPT on its jobset

The labels v_i, give a value to the jobs' positions in the schedule. Like the positions on a single machine, the processing time of a job scheduled last on a machine only contributes once to the objective value. The processing time of the second last job contributes twice and, in general, the l-th last job contributes l times to the objective value. The algorithm assigns to the longest job the best position. The second longest job gets the second best position and so on. This way it builds up an optimal solution to the problem. The following proof of optimality of this algorithm for the $Q||\sum C_j$ problem follows the proof given by Horowitz and Sahni [HS76].

Theorem 2.2. *The MFT algorithm produces an optimal schedule.*

Proof. We assume without loss of generality that ties are broken based on index. So in step 2 of the MFT algorithm the longest job with highest index is placed first and for SPT the job with lowest index is scheduled first.

Let μ be the schedule produced by the MFT algorithm. Suppose μ is not optimal. Then there exists some other schedule μ^* which is optimal and has sum of completion times less than that of μ. Assume, without loss of generality, that μ^* is an optimal schedule such that it has the lowest index highest index job that is scheduled on a different machine in μ. So there is no optimal schedule μ' such that

$$\max_{j \in J} \left\{ j \,\middle|\, \mu'_j \neq \mu_j \right\} < \max_{j \in J} \left\{ j \,\middle|\, \mu^*_j \neq \mu_j \right\}.$$

Let j be the job with highest index that is scheduled on different machines in μ^* and μ and let μ_j and μ^*_j denote the machines on which j is scheduled in μ and μ^* respectively. Consider the moment just before the MFT algorithm placed job j on machine μ_j and let v_{μ_j} and $v_{\mu^*_j}$ be the value labels of machines μ_j and μ^*_j at that moment. Since all jobs with higher index than j have the same place in μ and μ^* we know that $v_{\mu^*_j}$ is the value label corresponding to job j in schedule μ^*. So $v_{\mu^*_j} p_j$ is the contribution of job j in schedule μ^* and $v_{\mu_j} p_j$ is the contribution of job j in schedule μ. Since the MFT algorithm placed j on μ_j we know that $v_{\mu_j} \leq v_{\mu^*_j}$.

We distinguish between two cases. First suppose that the number of jobs on machine μ_j in schedule μ^* is less than v_{μ_j}. Then placing job j on machine μ_j instead of machine μ^*_j changes the contribution of job j to the objective value by at most

$$\left(v_{\mu_j} - v_{\mu^*_j} \right) p_j \leq 0$$

and does not increase the contribution of any other job to the objective value. So this change results in a schedule that is no worse than μ^* and has lower highest index job that is scheduled on a different machine than in μ, which is a contradiction to our assumption.

Now suppose the number of jobs on machine μ_j in schedule μ^* is v_{μ_j} or more. Let l be the job with the v_{μ_j}-th largest index on machine μ_j in schedule μ^*. Note that $l < j$, since otherwise there is at least one job with index larger than j that is scheduled different than in schedule μ. Now interchanging jobs j and l in schedule μ^* results in a change of the objective value of

$$\left(v_{\mu^*_j} p_l + v_{\mu_j} p_j \right) - \left(v_{\mu^*_j} p_j + v_{\mu_j} p_l \right) = \left(v_{\mu^*_j} - v_{\mu_j} \right) (p_l - p_j) \leq 0.$$

This interchange results in a schedule that is no worse than μ^* and has lower highest index job that is scheduled on a different machine than in μ. In both cases we have a contradiction to our assumption, so there can not exist such a schedule μ^* and the schedule μ must be an optimal schedule. $\square$

2.4 The scheduling game

A game is defined by the strategies of its players and utility functions for each of those players. Strategies of players may influence the utility of other players. In this thesis we treat the uniformly related machine scheduling game with n jobs and m machines. We denote the set of jobs with J and the set of machines with M. Each job $j \in J$ has its own agent. These agents are the players of the game. The strategy of an agent consists of choosing one of the machines for his job. In this thesis we only consider so called pure strategies, strategies which consist of one choice. This is in contrast to mixed strategies, where players may define a probability distribution over all their possible choices. A strategy profile, σ, consists of n strategies, played by the n agents. The disutility function for agent j for strategy profile σ, $u_j(\sigma)$, is equal to the completion time of job j.

To determine the completion time of a job it is necessary to know how the jobs are scheduled per machine. To achieve this, the machines have a local scheduling rule. These scheduling rules can be used to improve the price of anarchy of the game and therefore can be interpreted as coordination mechanisms [CKN04]. Since SPT is optimal on one machine, an interest for SPT as a local scheduling rule is obvious. Therefore we only use SPT as the local scheduling rule for each machine. Furthermore, we assume that all machines use the same rule to break ties between jobs with equal length. Without loss of generality we assume that for the MFT algorithm the same tie-breaking rule is used, both in step 2 and step 6, and we order the jobs such that the tie breaking rule is lowest index first. Knowing the local scheduling rule for each machine, the strategy profile σ also defines a schedule of the jobs on the machines. Since both in the optimal solution and the equilibrium solution the scheduling per machine is done in the same way, we will use the strategy profile σ to denote the corresponding schedule. So both the schedule σ and the strategy profile σ refer to the same thing. Furthermore for any schedule σ, σ_j refers to the strategy of job j within σ or the machine on which j is scheduled in σ.

Within the game each of the agents has its objective, but we want to measure one social objective. This social objective in our case is minimizing the sum of completion times of the jobs. The sum of completion times is, unlike other social objective functions, the sum of the utilities of the agents. Such a social objective function is also referred to as a utilitarian social objective function.

Summarizing:

- Job set: $|J| = n$
- Machine set: $|M| = m$
- Local scheduling rule: SPT
- Strategy profile of all jobs: $\sigma \in M^n$
- Strategy of a job j: $\sigma_j \in M$
- Utility of a job j (to be maximized):

$$u_j(\sigma) = -C_j(\sigma) = -\sum_{\substack{k \leq j \\ \sigma_k = \sigma_j}} \frac{p_k}{s_{\sigma_j}}$$

- Social objective (to be minimized):

$$\sum_{j \in J} C_j$$

2.5 Equilibrium solution algorithm

If ν is a equilibrium solution in a uniformly related machine scheduling game, with SPT as local scheduling rule, the Nash equilibrium property, (1.1), gives us

$$\sum_{\substack{k \leq j \\ \nu_k = \nu_j}} \frac{p_k}{s_{\nu_j}} \leq \sum_{\substack{k < j \\ \nu_k = \sigma_j}} \frac{p_k}{s_{\sigma_j}} + \frac{p_j}{s_{\sigma_j}} \qquad \forall \sigma_j \in M, \forall j \in J, \tag{2.1}$$

where ν_k denotes the machine chosen by job k in ν and σ_j denotes an arbitrary other choice for job j.

For uniformly related machine scheduling, the SPT algorithm is also known as the Ibarra-Kim algorithm. The algorithm was described by Ibarra and Kim in [IK77], as a heuristic for unrelated machine scheduling with makespan objective. The following two lemma's shows that exactly this algorithm constructs the equilibrium solutions, in which we are interested.

Lemma 2.3. *The SPT algorithm yields an equilibrium solution.*

Proof. Let ν be the schedule produced by the SPT algorithm. Consider the moment that the SPT algorithm schedules job j on machine ν_j. We know that, at that moment, the choice of ν_j minimizes the completion time of job j. But, since all jobs with smaller index than j are already scheduled when SPT schedules j, any job considered later than j does not influence the completion time of j. Since this is true for any job j, ν must be an equilibrium schedule. □

Lemma 2.4. *The SPT algorithm constructs all equilibrium solutions for uniformly related machine scheduling games, depending on how ties are broken.*

Heydenreich, Müller and Uetz prove Lemma 2.4 in [HMU07] for unrelated machines. We will give a proof for uniformly related machines based on this result.

Proof. We already know that any resulting schedule of the SPT algorithm is an equilibrium solution. So proving that any equilibrium solution is a realization of the SPT algorithm, proves the theorem.

Suppose ν is an arbitrary equilibrium schedule. With C_j being the completion time of job j, a non-decreasing order of C_j is exactly a non-decreasing order of processing times. If it is not, then two jobs k and k' exist with $k' > k$ and $C_{k'} < C_k$. So k and k' are not scheduled on the same machine. But if k would choose machine $\nu_{k'}$, instead of ν_k, it would be processed before k'. But then, since k is not longer than k', it must have a new completion time smaller than the old completion time of k' and therefore its new completion time is smaller than its old completion time. Thus ν was not an equilibrium solution.

Now if we successively take the shortest job j and schedule it on machine ν_j, this reproduces the equilibrium solution. But, at the same time, we choose for each consecutive job with smallest index, a machine for which its completion time is minimal. If ν_j is not a machine for which job j's completion time is minimal, then, since all jobs with smaller index than j already have been scheduled, there is a machine on which j's completion time is smaller than C_j, if j chooses that machine. Thus ν_j is not a best response for j and ν is not an equilibrium solution. So scheduling the jobs in non-decreasing order according to ν is exactly a realization of the SPT algorithm. Since this is true for any equilibrium solution ν, this proves Lemma 2.4. □

We see that the price of anarchy for $Q||\sum C_j$ games is exactly the ratio between the objective value from an SPT schedule and the objective value of the optimal solution for the $Q||\sum C_j$ scheduling model. Such a ratio is also known as the performance ratio of the SPT algorithm on $Q||\sum C_j$. While both the $Q||\sum C_j$ model and the SPT algorithm are long known, there are no known results (to us) on this performance ratio, before the results from game theoretic research done lately ([CGM10], [CQ10]). This may well be due to the fact that already an efficient optimal algorithm was known and no one was interested in approximation algorithms for the problem.

2.6 Robust price of anarchy

Different ways exist to prove upper bounds on the price of anarchy, one of which is through a smoothness argument as described by Roughgarden in [Rou09]. If we translate Roughgardens definition of smoothness to our scheduling game, then it says that the game is (λ, μ)-smooth if, for any two schedules σ and σ^*

$$\sum_{j=1}^{n} C_j(\sigma_j^*, \sigma_{-j}) \leq \mu \sum_{j=1}^{n} C_j(\sigma) + \lambda \sum_{j=1}^{n} C_j(\sigma^*), \tag{2.2}$$

for some $0 \leq \mu < 1$ and $\lambda \geq 0$. If this holds for any two schedules, then it also holds for any equilibrium schedule σ and optimal schedule σ^*. Since the left hand side of (2.2) sum over the completion time of job j in a schedule where j is moved from its equilibrium solution to another machine, the equilibrium solution must be less than the left hand side of (2.2). So, letting σ be the highest valued Nash equilibrium and σ^* be an optimal solution, from (2.2) we get

$$\begin{aligned}
&\sum_{j=1}^{n} C_j(\sigma) \leq \sum_{j=1}^{n} C_j(\sigma_j^*, \sigma_{-j}) \leq \mu \sum_{j=1}^{n} C_j(\sigma) + \lambda \sum_{j=1}^{n} C_j(\sigma^*) \\
\Leftrightarrow \quad & (1-\mu)\sum_{j=1}^{n} C_j(\sigma) \leq \lambda \sum_{j=1}^{n} C_j(\sigma^*) \\
\Leftrightarrow \quad & \text{POA} = \frac{\sum_{j=1}^{n} C_j(\sigma)}{\sum_{j=1}^{n} C_j(\sigma^*)} \leq \frac{\lambda}{1-\mu}.
\end{aligned}$$

In his paper Roughgarden shows that a smoothness argument not only suffices to prove the pure price of anarchy for a game, but it also proves the stronger *robust price of anarchy*, which he defines as

$$\inf \left\{ \frac{\lambda}{1-\mu} \middle| (\lambda, \mu) \text{ s.t. the game is } (\lambda, \mu)\text{-smooth} \right\}.$$

Furthermore Roughgarden shows that the robust price of anarchy is also gives the same upper bound on no-regret sequences, course correlated equilibriums and mixed Nash equilibriums.

2.7 Related Work

Recently two papers established bounds on the price of anarchy for scheduling models with sum of completion times objective. In [CQ10], Correa and Queranne show that the price of anarchy for the restricted uniformly related machine scheduling game with weighted sum of completion times objective ($RQ||\sum w_j C_j$) is equal to 4. Correa and Queranne use a generalized version of SPT as local scheduling rule, namely the *Weighted Shortest Processing Time First* rule (WSPT). This scheduling rule schedules jobs according to increasing $\frac{p_j}{w_j}$ ratio.

In their paper Correa and Queranne use two lemmas that do apply to $RQ||\sum w_j C_j$ games, but not to $Q||\sum C_j$ games. *The Essential Machines Lemma* says that for any instance of the game there is always an instance where all jobs are restricted to only two machines that attains the same price of anarchy. The idea of the proof is that it is possible to restrict every job to only the machines it is on in the optimal and the equilibrium solution. Then both these solutions will stay the same and thus the price of anarchy will stay the same. Furthermore Correa and Queranne show that there is an instance with highest price of anarchy that has $w_j = p_j$ for all jobs j. Of course both these lemmas do hold for $Q||\sum C_j$ games, but they both result in new problems that do not fit into the $Q||\sum C_j$ model. To be precise, the application of both lemmas results in exactly the the same set of instances of $RQ||\sum w_j C_j$ games that Correa and Queranne use to prove their result. Therefore this will not lead to any new result for $Q||\sum C_j$ games.

Correa and Queranne also construct instances, which price of anarchy approach 4 arbitrarily close. These instances have unit machine speeds and unit job lengths. The high price of anarchy is the direct result of the restricted sets of machines on which each job can be processed.

Cole, Gkatzelis and Mirrokni ([CGM10]) treat $R||\sum w_jC_j$ games with, among other coordination mechanisms, WSPT as local scheduling rule. They show that the upper bound on the price of anarchy of 4 also holds for the more general $R||\sum w_jC_j$ games. Their proof is in fact a $(2, \frac{1}{2})$-smoothness result, so they actually prove an upper bound on the robust price of anarchy. In Appendix C we follow this proof to find an upper bound of 4 for $Q||\sum C_j$. Furthermore Cole et al. prove a lower bound of 3 for the POA of $R||\sum C_j$. Their proof comes from a so-called game graph, where every machine is represented by a vertex and every job by an edge from the machine it is on in the optimal schedule to the machine it is on in the equilibrium solution. This cannot be done in the $Q||\sum C_j$ setting since it requires restricting a job to using only one of two machines, while other jobs are restricted to other machines.

Chapter 3

Instances with large price of anarchy

In this chapter we set out to find instances of the $Q||\sum C_j$ game, with high price of anarchy. First we derive properties of the game that narrow down the set of instances that can have maximal price of anarchy. After that we discuss some simple instances and compute their price of anarchy. In Section 3.3 we define a set of instances that results in a lower bound on the price of anarchy of $\frac{e}{e-1} \approx 1.5820$.

3.1 Reducing the set of instances

In this section we derive properties of the $Q||\sum C_j$ game that we use as tools for finding instances with high price of anarchy. In Chapter 4 we will also use these tools to find upper bounds on the price of anarchy.

The first reduction is based on the following lemma:

Lemma 3.1. *Any instance I may be rescaled such that the speed of machine 1 is set to 1, without changing the processing times $p_{ij} = \frac{p_j}{s_i}$ for pairs (i,j), $i \in M, j \in J$.*

Proof. Suppose the instance I has speed s_1 on machine 1. We multiply all machine speeds and all job lengths by a factor $\frac{1}{s_1}$. Machine 1 now has speed 1 and the processing time of any job j on any machine i is $\frac{1}{s_1} \cdot p_j / \left(\frac{1}{s_1} \cdot s_i\right) = \frac{p_j}{s_1}$. So for all j and i the processing time, $\frac{p_j}{s_i}$, remains the same. $\square$

As a consequence we may, without loss of generality, restrict to instances where the speed of machine 1 is 1. In a similar way we could change the length of one job instead of the speed of one machine. But, since we are only interested in preserving the price of anarchy and not the whole instance, we use the following somewhat stronger observation.

Lemma 3.2. *Multiplying the lengths of all jobs by a factor α does not affect the price of anarchy.*

Proof. Consider the MFT algorithm. Notice that multiplying all lengths by a factor α does not change the order in which the jobs are treated by the algorithm. So, since the MFT algorithm only considers the order of the jobs and not their exact length, it directly follows that all jobs are on the same machine as they would be without the change in length. So the objective value of the optimal solution is just α times the objective value before the change in length.

Now consider the SPT algorithm. If we proof that, for the SPT algorithm, the objective value also changes by exactly a factor α, we are done. Notice again that multiplying all lengths by a factor α does not change the order in which the jobs are treated by the algorithm. Let ν be the SPT schedule for the unchanged instance and ν' be the SPT schedule for the changed instance. Consider, for both instance, the moment just before job j is scheduled by SPT. It is easy to see that if all jobs with smaller index than j are, per job, on the same machine in ν and ν', job j is also scheduled on the same machine in both schedules and that its completion time in ν' is exactly α times its completion time in ν. But, since this holds for any job, ν and ν' must be the same schedule, up to a factor α, and the objective value of nu' is exactly α time the objective value of ν. □

From Lemmas 3.1 and 3.2 together, we assume without loss of generality that $s_1 = 1$ and $p_1 = 1$, unless stated otherwise.

Lemma 3.3. *A machine that is not used in an optimal solution will neither be used in an equilibrium solution with highest objective value.*

Proof. Suppose the lemma does not hold. Then, for some optimal solution μ and any equilibrium schedule with highest objective value ν, there is at least one machine that has no jobs scheduled on it in μ, but does have a job scheduled on it in ν.

Let i be a machine with the highest index that has no jobs scheduled on it in μ. We rescale the whole model to have $s_i = 1$.

Using value labels as in the MFT algorithm, a potential job on machine i would get value label 1. This means that, in the optimal solution, none of the other machines can have a job on a position with higher value. It immediately follows that no machine can have more jobs than its own speed rounded down. So, for the total number of jobs,

$$n \leq \sum_{l=i+1}^{m} \lfloor s_l \rfloor. \tag{3.1}$$

Suppose j is the longeest job such that $\nu_j = i$ and let $C_j(l)$ be its completion time if it would be scheduled on machine l. Then

$$C_j(l) \geq C_j(i) \geq p_j \quad \forall l.$$

If all machines $l' > i$ have at least $\lfloor s_{l'} \rfloor$ jobs scheduled on it in ν, we get

$$n \geq 1 + \sum_{l=i+1}^{m} \lfloor s_l \rfloor,$$

which is a contradiction with respect to (3.1). So we conclude that there must be at least one machine $l' > i$ that has at most $\lfloor s_{l'} \rfloor - 1$ jobs scheduled on it in ν.

Now, if we schedule j on l', j must be the $\lfloor s_{l'} \rfloor$-th largest job on l', since otherwise

$$C_j(l') = \leq \frac{p_j}{s_{l'}} + \sum_{\substack{k<j \\ \nu_k = l'}} \frac{p_k}{s_{l'}} < \frac{\lfloor s_l \rfloor \cdot p_j}{s_{l'}} \leq p_j \leq C_j(i),$$

which is a contradiction. But that means that j can be scheduled on machine l' without increasing the completion time job j itself or any other job. This results in an equilibrium schedule with less jobs scheduled on machines that are empty in μ, which is a contradiction. □

From lemma 3.3 we know that we may assume that for an instance with maximal price of anarchy all machines are used in the optimal solution.

Lemma 3.4. *There exists an instance with maximal price of anarchy in which the machines that are empty in the equilibrium solution all have the same speed.*

Proof. We know we may assume there are no machines that are empty in the optimal solution. So any empty machine in the equilibrium solution is used in the optimal solution. Let the highest speed of any machine that is empty in the equilibrium solution be s. Then, letting every empty machine have speed s, only improves the optimal solution, while the machines stay empty in the equilibrium solution, which therefore does not change. So this can only increase the price of anarchy. □

3.2 Simple instances

In this section we construct some instances for which we compute the price of anarchy. We start by looking at instances for which the number of machines is equal to the number of jobs and machines and jobs can be paired such that the speed of the machine equals the length of the job. So

$$\begin{aligned} n &= m \\ p_j &= s_j \qquad\qquad \forall j \in \{1, \ldots, n\}. \end{aligned}$$

We look at instances for which both in the optimal solution and the equilibrium solution each machine has only one job. Then in both the optimal and the equilibrium solution the completion time of a job is equal to it length divided by the speed of the machine it is on. When using the MFT algorithm to find the optimal solution each job is placed on the machine with the lowest value label. However, since no machine gets two jobs in the optimal solution, this is the same as picking the fastest machine that has no job placed on it. The order in which the jobs are placed on the machines is from longest first, so job j is scheduled on machine j. In the equilibrium solution we also have that no machine has two jobs scheduled on it, so only machines that are empty need to be considered to schedule a job on. Since the jobs are scheduled in SPT order and on the machine that gives them the lowest completion time, a job j is scheduled on machine $n - j + 1$.

Since, in the optimal solution, job j is scheduled on machine j and $s_j = p_j$, the objective function in the optimal solution is just the number of jobs, n. So the price of anarchy is just the average completion time in the equilibrium solution. Consider jobs j and $n - j + 1$. These jobs interchange machines when going from the optimal solution to the equilibrium solution. We can remove these two jobs with their corresponding machines without changing the completion time of any other job. Now let jobs k and $n - k + 1$ be the two jobs such that

$$C_k(\nu) + C_{n-k+1}(\nu) = \max_{j \in J} \left(C_j(\nu) + C_{n-j+1}(\nu) \right).$$

Then

$$\begin{aligned} \mathrm{POA}(I) &= \frac{\sum_{j \in J} C_j}{n} \\ &= \frac{1}{2} \frac{\sum_{j \in J} C_j}{n} + \frac{1}{2} \frac{\sum_{j \in J} C_{n-j+1}}{n} \\ &= \frac{1}{2} \frac{\sum_{j \in J} C_j + C_{n-j+1}}{n} \\ &\leq \frac{1}{2} \frac{\sum_{j \in J} C_k + C_{n-k+1}}{n} \\ &= \frac{C_k + C_{n-k+1}}{2}, \end{aligned} \tag{3.2}$$

which is the price of anarchy of an instance with only jobs k and $n - k + 1$ and their corresponding machines. Considering this we only have to look at instances with two machine and two jobs with $p_1 = s_1 = 1$ and $p_2 = s_2$. Then (3.2) becomes

$$\frac{1/s_2 + s_2}{2}.$$

Now it is easy to see that the maximum price of anarchy is $\frac{13}{12}$, which is for $s_2 = \frac{3}{2}$.

A somewhat more interesting set of instances to look at are those that have two machines, but with arbitrary processing times for the jobs. To keep the optimal and equilibrium solutions easy to determine we restrict the number of jobs to 3. So

$$m = 2$$
$$n = 3$$

For such an instance I, let $s_1 = 1$, $s_2 = s > 1$ and $p_1 = 1$. If $s < 2$ the optimal solution is $\mu = (\mu_1, \mu_2, \mu_3) = (2, 1, 2)$. If $2 \leq s < 3$ the optimal solution is

$(1, 2, 2)$. We need not consider $s \geq 3$ since then all jobs are on machine 2 in both the optimal and the equilibrium solution.

First suppose $s < 2$. In the equilibrium solution, job 1 is always on machine 2. So for $s < 2$ job 2 also needs to be on machine 2 otherwise the equilibrium solution is optimal. This gives us that

$$p_2 \geq \frac{1+p_2}{s} \Leftrightarrow p_2 \geq \frac{1}{s-1}.$$

After this there are two possibilities for job 3, namely either it has

$$p_3 < \frac{1+p_2}{s-1},$$

in which case it is on machine 1 in the equilibrium solution, or

$$p_3 \geq \frac{1+p_2}{s-1}$$

and then it is on machine 2. It is easy to see that both $C_1(\mu) \leq C_1(\nu)$ and $C_2(\mu) \leq C_2(\nu)$. So job 3 is the only job with completion higher in the equilibrium solution than in the optimal solution. Now let p_3 and p_3' be two different possible lengths of job 3 and $C_3(\nu)$, $C_3(\mu)$ and $C_3'(\nu)$, $C_3'(\mu)$ the corresponding completion times. Then, given p_2, if $\frac{C_3(\nu)}{C_3(\mu)} \leq \frac{C_3'(\nu)}{C_3'(\mu)}$ and $C_3(\nu) \leq C_3'(\nu)$, then

$$\frac{C_1(\nu)+C_2(\nu)+C_3(\nu)}{C_1(\mu)+C_2(\mu)+C_3(\mu)} \leq \frac{C_1(\nu)+C_2(\nu)+C_3'(\nu)}{C_1(\mu)+C_2(\mu)+C_3'(\mu)}.$$

From this and Lemma B.1 (see Appendix B), we see that if p_3 is such that job 3 is on machine 1, taking p_3 maximal gives the highest price of anarchy. This would mean that $p_3 = \frac{1+p_2}{s-1}$, but then we may assume $p_3 \geq \frac{1+p_2}{s-1}$. Then we can write the price of anarchy as

$$\text{POA}(I) = \frac{\frac{3+2p_2+p_3}{s}}{p_2 + \frac{2+p_3}{s}} = \frac{3+2p_2+p_3}{2+sp_2+p_3}.$$

Then, since $\frac{3+2p_2}{2+sp_2} \geq 1$ and using Lemma B.1, for the maximal price of anarchy we need p_3 to be minimal. So $p_3 = \frac{1+p_2}{s-1}$. This leaves us with

$$\text{POA}(I) = \frac{3+2p_2+\frac{1+p_2}{s-1}}{2+sp_2+\frac{1+p_2}{s-1}} = \frac{3s-2+(2s-1)p_2}{2s-1+(s^2-s+1)p_2}.$$

From Corollary B.2 we know that if

$$\frac{3s-2}{2s-1} \geq \frac{2s-1}{s^2-s+1} \qquad \forall 1 \leq s \leq 2,$$

then p_2 should be minimal. Now the above is equivalent to

$$\begin{aligned} & (3s-2)(s^2-s+1) \geq (2s-1)^2 \\ \Leftrightarrow \quad & (3s-2)(s^2-s+1)-(2s-1)^2 \geq 0 \\ \Leftrightarrow \quad & 3s^3-9s^2+9s-3 \geq 0. \end{aligned}$$

This is true for all s, $1 < s \leq 2$. So $p_2 = \frac{1}{s-1}$. Now we have $p_1 = 1, p_2 = \frac{1}{s-1}, p_3 = \frac{s}{(s-1)^2}$, and we get an expression for the price of anarchy that only depends on s:

$$\mathrm{POA}(I) = \frac{3+\frac{2}{s-1}+\frac{s}{(s-1)^2}}{2+\frac{s}{s-1}+\frac{s}{(s-1)^2}} = \frac{3(s-1)^2+2(s-1)+s}{2(s-1)^2+s(s-1)+s} = \frac{3s^2-3s+1}{3s^2-4s+2}.$$

This expression is maximized at $s = 1+\frac{\sqrt{3}}{3}$ for which the price of anarchy is $\frac{3}{2}\cdot\frac{2+\sqrt{3}}{3+\sqrt{3}} \approx 1.1830$.

Now suppose $2 \leq s < 3$. The optimal solution is $(1,2,2)$. For the equilibrium solution there are two possibilities. Since $s \geq 2$, both job 1 and job 2 are scheduled on machine 2. The two possible equilibrium solutions are $(2,2,1)$ and $(2,2,2)$. The first of these solutions occurs when $p_3 \leq \frac{1+p_2}{s-1}$. If $p_3 \geq \frac{1+p_2}{s-1}$ the other schedule is the equilibrium solution. Notice that when $p_3 = \frac{1+p_2}{s-1}$ both are an equilibrium solution.

Suppose $p_3 \leq \frac{1+p_2}{s-1}$. Then the objective value in the optimal solution equals

$$\mathrm{OPT}(I) = 1+\frac{2p_2+p_3}{s}.$$

For the equilibrium solution we have

$$\mathrm{NE}(I) = \frac{2+p_2}{s}+p_3.$$

This gives us

$$\mathrm{POA}(I) = \frac{2+p_2+sp_3}{s+2p_2+p_3}.$$

From Lemma B.1 and Corollary B.2, we see that, since $s \geq 2$, to maximize this expression, p_3 should be maximized, while p_2 should be minimized. This means that $p_2 = 1$ and $p_3 = \frac{2}{s-1}$. So

$$\mathrm{POA}(I) = \frac{2+1+s\frac{2}{s-1}}{s+2+\frac{2}{s-1}} = \frac{5s-3}{s^2+s},$$

which has its maximum at $s = 2$. Then the price of anarchy is equal to $\frac{7}{6} \approx 1.1667$.

Finally suppose $p_3 \geq \frac{1+p_2}{s-1}$, such that all jobs are on machine 2 in the equilibrium solution. Then for the price of anarchy we get

$$\mathrm{POA}(I) = \frac{3+2p_2+p_3}{s+2p_2+p_3}.$$

From Corollary B.2 we see that, to maximize this expression, in this case both p_2 and p_3 need to be minimal. So $p_2 = 1$ and $p_3 = \frac{2}{s-1}$, which is exactly the same as the above. So again the price of anarchy is $\frac{7}{6} \approx 1.1667$.

We conclude that, for instances with two machines and 3 jobs, the instance with the maximal price of anarchy has $s_1 = 1, s_2 = 1 + \frac{\sqrt{3}}{3}$ and $p_1 = 1, p_2 = \frac{1}{s_2-1}, p_3 = \frac{1+p_2}{s_2-1}$. This instance has price of anarchy $\frac{3}{2} \cdot \frac{2+\sqrt{3}}{3+\sqrt{3}} \approx 1.1830$.

We summarize the results from this section in the next theorem.

Theorem 3.5. *For $Q||\sum C_j$ game instances with two machines the price of anarchy is at least $\frac{3}{2} \cdot \frac{2+\sqrt{3}}{3+\sqrt{3}} \approx 1.1830$. Furthermore for instances with no more than 3 jobs this lower bound is tight.*

3.3 Instances with all jobs on the fastest machine in equilibrium

The idea to look at instances that have all jobs scheduled on the fastest machine in the equilibrium solution comes from the next lemma.

Lemma 3.6. *No equilibrium solution can have higher objective value than the schedule in which all jobs are scheduled in SPT order on the fastest machine. So*

$$\mathrm{NE}(I) \leq \sum_{j=1}^{n} \sum_{k=1}^{j} \frac{p_k}{s_m} = \sum_{j=1}^{n} (n-j+1) \frac{p_j}{s_m}. \tag{3.3}$$

Proof. Let C_j be the completion time of job j for an arbitrary equilibrium solution ν. Then we know that

$$C_j \leq \sum_{\substack{k<j \\ \nu_k = i}} \frac{p_k}{s_i} + \frac{p_j}{s_i} \qquad \forall i, j.$$

So in particular:

$$\begin{aligned} C_j &\leq \sum_{\substack{k<j \\ \nu_k = m}} \frac{p_k}{s_m} + \frac{p_j}{s_m} \\ &\leq \sum_{k<j} \frac{p_k}{s_m} + \frac{p_j}{s_m} \\ &= \sum_{k=1}^{j} \frac{p_k}{s_m} \qquad \forall j. \end{aligned}$$

Now, summing over all j gives the desired result

$$\sum_{j=1}^{n} C_j \leq \sum_{j=1}^{n} \sum_{k=1}^{j} \frac{p_k}{s_m}.$$

□

Lemma 3.6 clearly suggests that instances which have all jobs on the fastest machine in the equilibrium solution are interesting to look at. However it does not say anything about whether these instances will contain an instance which has the highest price of anarchy overall.

From Lemma 3.4 we know that we may assume that every machine that is empty in the equilibrium solution has speed 1. We let s be the speed of the fastest machine and all other machines have speed 1.

When all jobs are on the fastest machine in the equilibrium solution, the number of slow machines does not change the equilibrium. However it does change the optimal solution. From the MFT algorithm it is easy to see that, in the optimal solution, the first $\lfloor s_i \rfloor$ jobs are scheduled on the fast machine. The remaining jobs are scheduled on the slow machines until every slow machine has a job assigned to it. The contribution of the jobs that are remaining after that is higher than the length of the job. From this it is easy to see that the instances with the highest price of anarchy have

$$\sum_{i=1}^{m} \lfloor s_i \rfloor \geq n.$$

It directly follows that the optimal solution is minimal when there are at least $n - \lfloor s \rfloor$ slow machines.

Consider a job j. When all preceding jobs are on the fastest machine in the equilibrium solution, we get for the processing time of job j

$$\begin{aligned}
& & p_j &\geq \sum_{k=1}^{j} \frac{p_k}{s} \\
\Leftrightarrow & & \left(1 - \frac{1}{s}\right) p_j &\geq \sum_{k=1}^{j-1} \frac{p_k}{s} \\
\Leftrightarrow & & p_j &\geq \frac{s}{s-1} \sum_{k=1}^{j-1} \frac{p_k}{s} = \frac{1}{s-1} \sum_{k=1}^{j-1} p_k
\end{aligned} \tag{3.4}$$

So if all jobs satisfy (3.4), then all jobs are on the fastest machine in the equilibrium solution.

It is easy to see that when $s = 2$, inequality (3.4) becomes

$$p_j \geq \sum_{k=1}^{j-1} p_k.$$

With this in mind we look at the following set of instances.

Instance 1. Let n be the number of jobs, two of which have length 1 and the remaining $n-2$ jobs have length $p_j = \sum_{k=1}^{j-1} p_k$, for all j, $3 \leq j \leq n$. Let $m = n - 1$ be the number of machines, of which one has speed 2 and the rest of the machines has speed 1.

Easy calculations show that for Instance 1

$$p_j = \begin{cases} 1 & \text{if } j \in \{1,2\} \\ 2^{j-2} & \text{otherwise} \end{cases}$$

The lengths of the jobs satisfy (3.4), so the schedule with all jobs on the fastest machine is the equilibrium solution with highest objective value. An optimal solution is when the two longest jobs are on the fast machine and all other jobs are on a machine with speed 1.

Theorem 3.7. *The price of anarchy of Instance 1 goes to $\frac{4}{3}$ as n goes to infinity.*

Proof. Let I denote Instance 1. For $n \geq 4$, the objective value in the optimal solution (OPT(I)) is

$$\text{OPT}(I) = \sum_{j=1}^{n-2} p_j + \frac{p_{n-1}}{2} + \frac{p_{n-1} + p_n}{2} = p_{n-1} + p_{n-1} + \frac{p_n}{2} = 2 \cdot 2^{n-3} + \frac{2^{n-2}}{2} = 3 \cdot 2^{n-3}$$

and the objective value in the equilibrium solution (NE(I)) is

$$\begin{aligned} \text{NE}(I) &= \sum_{j=1}^{n} \sum_{k=1}^{j} \frac{p_k}{2} = \frac{p_1}{2} + \frac{p_1 + p_2}{2} + \sum_{j=3}^{n} \frac{p_{j+1}}{2} \\ &= \frac{p_1}{2} + \frac{p_1 + p_2}{2} + \sum_{j=3}^{n} \frac{2^{j-1}}{2} = \frac{p_1}{2} + p_2 + \sum_{j=3}^{n} 2^{j-2} \\ &= \sum_{j=1}^{n} p_j - \frac{p_1}{2} = 2^{n-1} - \frac{1}{2} = 4 \cdot 2^{n-3} - \frac{1}{2}. \end{aligned}$$

Now the price of anarchy (POA(I)) is

$$\text{POA}(I) = \frac{\text{NE}(I)}{\text{OPT}(I)} = \frac{4 \cdot 2^{n-3} - \frac{1}{2}}{3 \cdot 2^{n-3}} = \frac{4}{3} - \frac{1}{3 \cdot 2^{n-2}},$$

which indeed goes to $\frac{4}{3}$ as n goes to infinity. □

The main theorem of this chapter uses instances like Instance 1, but with variable speed for the fastest machine.

Theorem 3.8. *The price of anarchy for uniformly related machine scheduling games with sum of completion times objective is no less than $\frac{e}{e-1} \approx 1.5820$.*

Proof. Consider an instance I with all n jobs on the fastest machine in the equilibrium solution. Let the speed of the fastest machine be s. For ease of calculations let s be integer. The speed of all other machines is 1. Let there be $m = n - s + 1$ machines. This gives

$$\sum_{i=1}^{m} s_i = n.$$

Let

$$x = \frac{s}{s-1}$$

and define the lengths of the jobs as

$$p_j = \begin{cases} 1 & \text{if } 1 \leq j \leq s \\ x^{j-s} & \text{if } s+1 \leq j \leq n \end{cases} .$$

The first s jobs automatically satisfy (3.4) and, since

$$\frac{1}{s-1}\sum_{j=1}^{k} p_j = \frac{1}{s-1}p_k + \frac{1}{s-1}\sum_{j=1}^{k-1} p_j \leq \frac{1}{s-1}p_k + p_k = \frac{s}{s-1}p_k,$$

if the preceding job j, $j \in \{s, \ldots, n-1\}$ fulfills (3.4), then the succeeding job, $j+1$, does too.

Note that, when we take the sum of k summands of size x^j, $j \in \{1, \ldots, k\}$, we get

$$\begin{aligned}
\sum_{j=0}^{k} x^j &= \sum_{j=0}^{k} \left(\frac{s}{s-1}\right)^j \\
&= \frac{\left(\frac{s}{s-1}\right)^{k+1} - 1}{\left(\frac{s}{s-1}\right) - 1} \\
&= (s-1)\left(\left(\frac{s}{s-1}\right)^{k+1} - 1\right) \\
&= (s-1)\left(\frac{s}{s-1}\right)^{k+1} - (s-1) = (s-1)x^{k+1} - (s-1) \\
&= s\left(\frac{s}{s-1}\right)^{k} - (s-1) = sx^k - (s-1).
\end{aligned}$$

In the optimal solution the s longest jobs are on the fastest machine. All other jobs are on a slow machine. So the objective value in the optimal solution is:

$$\begin{aligned}
\mathrm{OPT}(I) &= \sum_{j=1}^{s-1} p_j + \sum_{j=s}^{n-s} p_j + \sum_{j=n-s+1}^{n} \sum_{k=n-s+1}^{j} \frac{p_j}{s} \\
&= \sum_{j=1}^{s-1} p_j + \sum_{j=s}^{n-s} x^{j-s} + \sum_{j=n-s+1}^{n} \sum_{k=n-s+1}^{j} \frac{x^{k-s}}{s} \\
&= \sum_{j=1}^{s-1} 1 + \sum_{j=0}^{n-2s} x^j + \sum_{j=n-s+1}^{n} \frac{1}{s} \left(\sum_{k=0}^{j-s} x^k - \sum_{k=0}^{n-2s} x^k \right) \\
&= s-1+(s-1)x^{n-2s+1} - (s-1) + \sum_{j=n-s+1}^{n} \left(x^{j-s} - x^{n-2s+1} \right) \\
&= (s-1)x^{n-2s+1} + \sum_{j=n-2s+1}^{n-s} x^j - \sum_{j=n-s+1}^{n} x^{n-2s} \\
&= (s-1)x^{n-2s+1} + (s-1)x^{n-s+1} - (s-1)x^{n-2s+1} - sx^{n-2s} \\
&= (s-1)x^{n-s+1} - (s-1)x^{n-2s+1}. \qquad (3.5)
\end{aligned}$$

In the equilibrium solution, ν, all jobs are on the fastest machine. The completion time of the first $s-1$ jobs is

$$C_j(\nu) = \frac{j}{s} \qquad \forall j \leq s-1$$

and for the s-th job this is 1. For every next job we have:

$$\begin{aligned}
C_j(\nu) &= \sum_{k=1}^{j} \frac{p_k}{s} \\
&= \frac{s-1}{s} + \sum_{k=s}^{j} \frac{x^{j-s}}{s} \\
&= \frac{s-1}{s} + \sum_{k=0}^{j-s} \frac{x^j}{s} \\
&= \frac{s-1}{s} + x^{j-s} - \frac{s-1}{s} \\
&= x^{j-s} = p_j.
\end{aligned}$$

So in the equilibrium solution every job, except for the first $s-1$, has completion time equal to its length.

This makes computing the value of the objective function in the equilibrium solution easy:

$$\begin{aligned}
\mathrm{NE}(I) &= \sum_{j=1}^{s-1} \frac{j}{s} + \sum_{j=s}^{n} x^{j-s} \\
&= \frac{s(s-1)}{2s} + \sum_{j=0}^{n-s} x^{j} \\
&= \frac{(s-1)}{2} + (s-1)x^{n-s+1} - (s-1) \\
&= (s-1)x^{n-s+1} - \frac{(s-1)}{2}. \qquad (3.6)
\end{aligned}$$

Combining (3.5) and (3.6) gives us the price of anarchy:

$$\begin{aligned}
\mathrm{POA}(I) &= \frac{(s-1)x^{n-s+1} - \frac{(s-1)}{2}}{(s-1)x^{n-s+1} - (s-1)x^{n-2s+1}} \\
&= \frac{x^{n-s+1} - \frac{1}{2}}{x^{n-s+1} - x^{n-2s+1}} \\
&= \frac{x^{s} - \frac{1}{2}x^{-(n-2s+1)}}{x^{s} - 1} \\
&= \frac{\left(\frac{s}{s-1}\right)^{s} - \frac{1}{2}\left(\frac{s}{s-1}\right)^{-(n-2s+1)}}{\left(\frac{s}{s-1}\right)^{s} - 1}. \qquad (3.7)
\end{aligned}$$

Now, if we let n go to infinity, (3.7) becomes:

$$\lim_{n\to\infty} \mathrm{POA}(I) = \frac{\left(\frac{s}{s-1}\right)^{s}}{\left(\frac{s}{s-1}\right)^{s} - 1} \qquad (3.8)$$

and letting s also go to infinity, (3.8) goes to $\frac{e}{e-1}$, which is approximately 1.5820 $\square$

Chapter 4

Upper bounds on the price of anarchy

The instances from Theorem 3.8 imply a lower bound on the a price of anarchy of $\frac{e}{e-1} \approx 1.5820$. This chapter treats a couple of results leading to upper bounds on the price of anarchy.

4.1 Known bounds

Upper bounds on the price of anarchy for more general scheduling models automatically provide us with upper bounds for the uniformly related machine scheduling model. Although lower bounds for such models do not say anything about lower bounds for our model, they do provide useful insight. Namely, if we could prove an upper bound for our problem which is strictly less than a lower bound for a more general problem, we may conclude that our model has a strictly lower price of anarchy. Therefore it is interesting to look at both upper and lower bounds for more general models from other literature. The work of Cole, Gkatzelis and Mirrokni ([CGM10]) and Correa and Queyranne ([CQ10]) that we already discussed in section 2.7 provide both upper and lower bounds that also apply to the $Q||\sum C_j$ game. The lowest upper bound on the $Q||\sum C_j$ game, that follows directly from these results, is 4.

4.2 Unit speeds and unit lengths

In this section we prove that, when we look at the more specific models, with either jobs with unit length or parallel machine instances, equilibrium solutions are also optimal solutions. So for these instances the price of anarchy is equal to 1.

Theorem 4.1. *Equilibrium solutions for instances with unit job lengths are optimal.*

Proof. An instance with unit job lengths has $p_j = 1$ for all $j \in J$. So any ordering of the jobs is an SPT order and all jobs are interchangeable without effecting the value of the objective. Suppose both the SPT algorithm, in step 1, and the MFT algorithm, in step 2, choose the job with lowest index. Let ν be the schedule resulting from the SPT algorithm and let $C_k(\nu)$ be the completion time of job k within that schedule. Finally let $J_i(\nu)$ be the set of all jobs that are scheduled on machine i by the SPT algorithm. Then for $C_k(\nu)$ we have:

$$\begin{aligned} C_k(\nu) &= \min_{i\in M} \sum_{\substack{j<k \\ j\in J_i(\nu)}} \frac{p_j}{s_i} + \frac{p_k}{s_i} \\ &= \min_{i\in M} \sum_{\substack{j<k \\ j\in J_i(\nu)}} \frac{1}{s_i} + \frac{1}{s_i} \\ &= \min_{i\in M} \frac{|\{j|j<k,\, j\in J_i(\nu)\}|+1}{s_i} \qquad (4.1) \end{aligned}$$

This means that the algorithm schedules a job k on the machine with the smallest ratio (4.1), which is just the number of jobs on the machine plus one divided by the speed of that machine.

Consider the iteration in which the MFT algorithm adds job k to the job set of a machine, let $v(k)$ be the value label of that machine just before the current iteration. Let $J_i(\mu)$ be the set of jobs that the MFT algorithm schedules on machine i. The following holds:

$$\begin{aligned} v(k) &= \min_{i\in M} \sum_{\substack{j<k \\ j\in J_i(\mu)}} \frac{1}{s_i} + \frac{1}{s_i} \\ &= \min_{i\in M} \frac{|\{j|j<k,\, j\in J_i(\mu)\}|+1}{s_i} \end{aligned}$$

So in fact the MFT algorithm chooses for job k exactly the same machine as SPT does as long as both algorithms have the same number of jobs on each machine. Since initially the number of jobs on each machine is zero for both algorithms, both algorithms choose exactly the same number of jobs for each machine. So any equilibrium schedule is also an optimal solution for instances with unit job lengths. □

Theorem 4.2. *Equilibrium solutions for parallel machine scheduling instances are optimal.*

In section 5.3 of [Pin08] a intuitive proof of Theorem 4.2 is given. Here we give a full formal proof of the theorem.

Proof. Assume that $\frac{n}{m}$ is an integer number. If this is not the case add dummy jobs with length 0 such that it is an integer number.

Consider the MFT algorithm. At any iteration, the value labels of any machine equals the number of jobs already placed on that machine plus one. So the machine with the lowest value label is simply the machine with the least jobs placed on it. If there are m machines, the m longest jobs will be last on a machine, the next m longest jobs will be second last on a machine and so on. So any job k will have $\lfloor \frac{n-k}{m} \rfloor$ jobs scheduled after it on its machine.

Now consider the SPT algorithm. If we know the place of a job on a machine we know its contribution to the sum of completion times. So if the SPT algorithm schedules all jobs on the same place on a machine as the MFT algorithm, the two schedules have the same objective value. We will use a proof by mathematical induction on the number of iterations of the SPT algorithm.

Note that, since all speeds are equal, only the starting time of the job to be scheduled in step 2 of the SPT algorithm is decisive for its completion time. So the machine it will be scheduled on is the machine with the smallest load.

Base step: For the m jobs with lowest index SPT will always choose an empty machine. Without loss of generality assume that the first m jobs are scheduled such that job j is scheduled on machine i and $j = i$.

Induction hypothesis: For $1 \leq j \leq k-1$ every job j is scheduled on the $\left(\frac{j}{m} - \lfloor \frac{j}{m} \rfloor\right)$-th machine.

Induction step: Say $k = x \cdot m + l$, for $1 \leq l \leq m$. Then, every machine $1, \dots, l-1$ has $x+1$ jobs with smaller index than k scheduled on it, and every machine $l, \cdots, m$ has x jobs with smaller index than k scheduled on it. Let J_i be the set of jobs that SPT schedules on machine i. From the induction hypothesis we have, for machines $1, \dots, l-1$

$$\sum_{\substack{j<k \\ j \in J_i}} p_j = \sum_{z=0}^{x} p_{z \cdot m+i} \qquad \forall i \in \{1, \dots, l-1\}.$$

For machines $l, \dots, m$ we have:

$$\sum_{\substack{j<k \\ j \in J_i}} p_j = \sum_{z=0}^{x-1} p_{z \cdot m+i} \qquad \forall i \in \{l, \dots, m\}.$$

For any two jobs j and j' we know that $p'_j \geq p_j$, if and only if $j' \geq j$. So for any machine $i \in \{l, \dots, m\}$:

$$\begin{aligned}
\sum_{\substack{j<k\\ j\in J_i}} p_j &= \sum_{z=0}^{x-1} p_{z\cdot m+i}\\
&\leq \sum_{z=0}^{x-1} p_{z\cdot m+m}\\
&= \sum_{z=0}^{x-1} p_{(z+1)\cdot m}\\
&= \sum_{z=1}^{x} p_{z\cdot m}\\
&\leq \sum_{z=1}^{x} p_{z\cdot m+i'}\\
&\leq \sum_{z=0}^{x} p_{z\cdot m+i'} && \forall i' \in \{1,\ldots,l-1\}
\end{aligned}$$

and of course:

$$\sum_{\substack{j<k\\ j\in J_i}} p_j = \sum_{z=0}^{x-1} p_{z\cdot m+i} \leq \sum_{z=0}^{x-1} p_{z\cdot m+i'} \qquad \forall i' > i.$$

So the machine with the earliest starting time for job k is the machine with lowest index from $\{l,\ldots,m\}$. Which is indeed machine l.

So by use of mathematical induction we find that each job $k = x\cdot m + l$ is scheduled on machine l. Note that job k is scheduled $(x+1)$-st on that machine. So for the number of jobs scheduled after job k we have

$$\sum_{\substack{j>k\\ k\in J_l}} 1 = |J_l| - (x+1) = \frac{n}{m} - \left(\left\lfloor\frac{k}{m}\right\rfloor + 1\right),$$

which is equal to $\left\lfloor\frac{n-k}{m}\right\rfloor$. So in both schedules each job has the same number of jobs scheduled after it and thus the contribution of each job is also the same in both schedules. □

Knowing that, for both instances with unit length jobs and instances in a parallel machine environment, the price of anarchy is 1, the problem we are dealing with is one of the simplest scheduling game models, with sum of completion time objective, which has price of anarchy greater than 1.

4.3 Upper bounds on the price of anarchy for the $Q||\sum C_j$ game

We used different methods to find upper bounds on the price of anarchy. At first Instance 1 was the only instance we had with high price of anarchy, so we tried to compare arbitrary instances to Instance 1. Lemma 4.3 shows such a comparison result.

Another approach is to compute lower bounds on the objective value of an optimal solution and upper bounds on the objective value in an equilibrium solution. Combining these bounds directly leads to an upper bound on the price of anarchy. In section 4.3.2 we compute lower bounds on the optimal solution. Together with the upper bounds for equilibrium solutions that we show in section 4.3.3, we combine these in section 4.3.4 to get upper bounds on the price of anarchy.

Our final method uses the Nash equilibrium property (1.1) to construct new schedules, which we use to bound the objective value of the equilibrium solution. We prove our main result, an upper bound of 3 on the price of anarchy, with this method.

4.3.1 Comparing instances

Since we already know an instance which we think has a high price of anarchy, we can try to compare other instances to that instance. If we succeed to proof $\text{POA}(I) \leq \alpha\text{POA}(I^*)$ for some specific instance I^* and I from a specified set of instances and $\alpha \in \mathbb{R}$, we get a constant upper bound for instances from that set.

The idea would be to begin with the instance I and through several transformations get to instance I^*. If then each step does not decrease the price of anarchy this proves $\text{POA}(I) \leq \text{POA}(I^*)$. Transformation that decrease the price of anarchy may be used as long as their product is no less than $\frac{1}{\alpha}$.

Lemma 4.3 is an example of a basic comparison result for instances with $m-1$ machine with speed 1 and one machine with speed 2 and jobs that have length equal to a power of 2.

Lemma 4.3. *Let I be a uniformly related machine scheduling game with two machine speeds, 1 and 2, $n-1$ machines, of which one has speed 2, and $m+1$ jobs with processing times, $p_j \in \left\{2^k | k \in \mathbb{N}\right\}$. Then there exists an instance I' with all jobs on the fastest machine in the equilibrium solution, with price of anarchy greater of equal to that of I.*

Proof. If I has all job on the fastest machine in the equilibrium solution we are done. So assume I has at least one job on a slow machine. Since an empty slow machine is always available, by definition $C_i(\nu) \leq p_i$, $\forall i \in J$. So for any job j that is not on the fastest machine in the optimal solution, $C_j(\nu) \leq C_j(\mu)$. Suppose such a job j is on a slow machine in the equilibrium solution. The length of job j does not influence the completion time of any other job in either

ν or μ. Then, since by definition $\sum_{i=1}^n C_i(\mu) \leq \sum_{i=1}^n C_i(\nu)$, removing j gives us

$$\mathrm{POA}(I') = \frac{\sum_{i=1}^n C_i(\nu) - p_j}{\sum_{i=1}^n C_i(\mu) - p_j} \geq \frac{\sum_{i=1}^n C_i(\nu)}{\sum_{i=1}^n C_i(\mu)}.$$

So j can be removed without lowering the price of anarchy. We assume that all of the $n-2$ shortest jobs are on the fast machine in the equilibrium solution.

Note that for $s = 2$, in the proof of Theorem 3.8, (3.8) equals $\frac{4}{3}$ and instances from the proof satisfy all constraints. So proving that, for any instance that does not have all jobs on the fastest machine in the equilibrium solution, the price of anarchy is less than $\frac{4}{3}$ is enough to prove the lemma.

Consider the 2 longest jobs. These are the only jobs that can have a higher completion time in the equilibrium solution than in the optimal solution. Their contribution to the objective in the optimal solution is

$$C_{n-1}(\mu) + C_n(\mu) = \frac{p_{n-1}}{2} + \frac{p_{n-1} + p_n}{2} = p_{n-1} + \frac{p_n}{2}.$$

Note that if job j is the shortest job that is not on the fastest machine in the equilibrium solution then

$$p_j < \sum_{i=1}^{j-1} p_i \leq 2p_{j-1}$$

and thus, since $p_i \in \left\{2^k | k \in \mathbb{N}\right\}$, $\forall i$, $p_j = p_{j-1}$.

We distinguish between three cases. Either both jobs n and $n-1$ are not on the fast machine, only job n is not on the fast machine, or only job $n-1$ is not on the fast machine. First suppose both jobs n and $n-1$ are not on the fast machine. Then $p_n = p_{n-1} = p_{n-2}$ and

$$\mathrm{POA}(I) = \frac{\sum_{i=1}^{n-2} C_i(\nu) + 2p_{n-2}}{\sum_{i=1}^{n-2} p_i + \frac{3}{2}p_{n-2}} \leq \frac{\sum_{i=1}^{n-2} p_i + 2p_{n-2}}{\sum_{i=1}^{n-2} p_i + \frac{3}{2}p_{n-2}} = \frac{\sum_{i=1}^{n-3} p_i + 3p_{n-2}}{\sum_{i=1}^{n-3} p_i + \frac{5}{2}p_{n-2}},$$

which is less than $\frac{6}{5}$.

Now suppose only job n is not on the fast machine. Then $p_n = p_{n-1}$ and $p_{n-1} \geq \sum_{i=1}^{n-2} p_i$. For the price of anarchy we get:

$$\mathrm{POA}(I) = \frac{\sum_{i=1}^{n-2} C_i(\nu) + \sum_{i=1}^{n-2} \frac{p_i}{2} + \frac{3}{2}p_{n-1}}{\sum_{i=1}^{n-2} p_i + \frac{3}{2}p_{n-1}} \leq \frac{\sum_{i=1}^{n-2} p_i + 2p_{n-1}}{\sum_{i=1}^{n-2} p_i + \frac{3}{2}p_{n-1}} < \frac{4}{3},$$

where the strict inequality holds, because if $\sum_{i=1}^{n-2} p_i = 0$, there would be only two jobs and the price of anarchy would be 1.

Finally suppose that job $n-1$ is the only job that is not on the fast machine. Then $p_{n-1} = p_{n-2}$ and $p_n \geq \sum_{i=1}^{n-2} p_i$. Notice also that if their are only three jobs with $p_1 = p_2$ the price of anarchy is 1, so assume $n > 3$. Then we also know that $p_n > p_{n-2}$ and thus $p_n \geq 2p_{n-2}$. Using this we see that

$$\mathrm{POA}(I) = \frac{\sum_{i=1}^{n-2} C_i(\nu) + \sum_{i=1}^{n-2} \frac{p_i}{2} + \frac{1}{2}p_n + p_{n-2}}{\sum_{i=1}^{n-2} p_i + \frac{1}{2}p_n + p_{n-2}} \leq \frac{\sum_{i=1}^{n-2} p_i + \frac{1}{2}p_n + 2p_{n-2}}{\sum_{i=1}^{n-2} p_i + \frac{1}{2}p_n + p_{n-2}}$$

and, since $p_n \geq 2p_{n-2}$, we have

$$\frac{\sum_{i=1}^{n-2} p_i + \frac{1}{2}p_n + 2p_{n-2}}{\sum_{i=1}^{n-2} p_i + \frac{1}{2}p_n + p_{n-2}} \leq \frac{\sum_{i=1}^{n-3} p_i + 4p_{n-2}}{\sum_{i=1}^{n-3} p_i + 3p_{n-2}} < \frac{4}{3}.$$

This completes the proof. □

Notice that, even if Lemma 4.3 can be extended to arbitrary s and more than one fast machine, we still have not proved that the instances from Theorem 3.8 have the highest price of anarchy for machine with two speeds.

4.3.2 Lower bounds on the optimal solution

Lemma 4.4. *For any uniformly related machine instance I*

$$\mathrm{OPT}(I) \geq \frac{\sum_{j=1}^{n}(n-j+1)p_j}{\sum_{i=1}^{m} s_i} = \frac{\sum_{j=1}^{n}\sum_{k=1}^{j} p_k}{\sum_{i=1}^{m} s_i}. \tag{4.2}$$

Proof. First we note that the right hand side of (4.2) is in fact the objective value of an SPT schedule on one machine with speed $\sum s_i$.

Let φ be a reordering of the jobs such that it corresponds to the order in which jobs are completed in the optimal solution. So in the optimal solution

$$C_{\varphi_1} \leq C_{\varphi_2} \leq \ldots \leq C_{\varphi_{n-1}} \leq C_{\varphi_n}.$$

Then we know that at time C_{φ_j} a load of $\sum_{k=1}^{j} p_{\varphi_k}$ must have been completed by the machines. The total capacity of the machines is $\sum_{i=1}^{m} s_i$ per unit of time. So completing the load of $\sum_{k=1}^{j} p_{\varphi_k}$ takes at least $\sum_{k=1}^{j} p_{\varphi_k} / \sum_{i=1}^{m} s_i$ units of time. So for the optimal schedule we have

$$C_{\varphi_j} \geq \frac{\sum_{k=1}^{j} p_{\varphi_k}}{\sum_{i=1}^{m} s_i}.$$

Summing over all j then gives

$$\begin{aligned}\sum_{j=1}^{n} C_{\varphi_j} &\geq \frac{\sum_{j=1}^{n}\sum_{k=1}^{j} p_{\varphi_k}}{\sum_{i=1}^{m} s_i} \\ &= \frac{\sum_{j=1}^{n}(n-j+1)p_{\varphi_j}}{\sum_{i=1}^{m} s_i}.\end{aligned}$$

This is the objective value of some schedule on the single machine instance with speed $\sum s_i$. And, since SPT is the optimal schedule for the single machine instance, we get

$$\mathrm{OPT}(I) \geq \frac{\sum_{k=1}^{n}(n-k+1)p_k}{\sum_{i=1}^{m} s_i},$$

which is the desired result. □

Lemma 4.5. *For any uniformly related machine instance I*

$$\mathrm{OPT}(I) \geq \frac{\sum_{j=1}^{n} p_j}{s_m}. \tag{4.3}$$

Proof. The best possible completion time for a job j in any schedule is when it is scheduled first on the fastest machine. Its completion time then, is $\frac{p_j}{s_m}$. So no schedule can be better than when all jobs have this completion time, which gives us

$$\mathrm{OPT}(I) \geq \frac{\sum_{j=1}^{n} p_j}{s_m}$$

as an upper bound on the optimal solution. □

Lemma 4.6. *Let I be a uniformly related machine scheduling instance, with job set J. Let I_{Ps} be the parallel machine scheduling instance with the same job set J and the same number of machines as I, but with the speeds of the machines set to s_m. Let μ be the optimal schedule for I and μ_{Ps} be the SPT schedule for I_{Ps}. Then*

$$\sum_{j \in J} C_j^I(\mu) \geq \sum_{j \in J} C_j^{I_{Ps}}(\mu_{Ps}), \tag{4.4}$$

where $C_j^I(\sigma)$ is the completion time of job j for instance I with schedule σ.

Proof. Note that μ_{Ps} is both an equilibrium and an optimal schedule for I_{Ps}. When we keep the optimal schedule on I and only speed up each machine to s_m, this can not increase the completion time of any job. This is then a schedule on the parallel machine instance, which has lower sum of completion times. Since μ_{Ps} is the optimal schedule for I_{Ps}, this can only improve the schedule even further and thus will never have higher objective value than μ on I. Thus

$$\sum_{j \in J} C_j^I(\mu) \geq \sum_{j \in J} C_j^{I_{Ps}}(\mu_{Ps}).$$

□

4.3.3 Upper bounds on the equilibrium solution

Lemma 4.7. *For any uniformly related machine instance, I, and equilibrium solution, ν,*

$$\sum_{j \in J} C_j(\nu) \leq \underbrace{\frac{\sum_{k=1}^{n}(n-k+1)p_k}{\sum_{i=1}^{m} s_i}}_{(1)} + \underbrace{\frac{\sum_{j=1}^{n} p_j}{\sum_{i=1}^{m} s_i/(m-1)}}_{(2)}. \tag{4.5}$$

Proof. We know that in the equilibrium solution

$$C_j(\nu) \leq \frac{\sum_{k \in J_i, k<j} p_k + p_j}{s_i} \qquad \forall j, i.$$

So

$$s_iC_j(\nu) \leq \sum_{k\in J_i, k<j} p_k + p_j \qquad \forall j, i.$$

Now summing over i gives

$$C_j(\nu) \leq \frac{\sum_{k<j} p_k}{\sum_{i=1}^m s_i} + \frac{mp_j}{\sum_{i=1}^m s_i} = \frac{\sum_{k\leq j} p_k}{\sum_{i=1}^m s_i} + \frac{(m-1)p_j}{\sum_{i=1}^m s_i} \qquad \forall j.$$

Now summing over j yields

$$\sum_{j\in J} C_j(\nu) \leq \underbrace{\frac{\sum_{k=1}^n (n-k+1)p_k}{\sum_{i=1}^m s_i}}_{(1)} + \underbrace{\frac{\sum_{j=1}^n p_j}{\sum_{i=1}^m s_i/(m-1)}}_{(2)}.$$

□

Lemma 4.8. *Let I be a uniformly related machine scheduling instance, with job set J. Let I_P be the parallel machine scheduling instance with the same job set J and the same number of machines as I, but with machines speed equal to 1. Let ν be the equilibrium schedule for I and ν_P be the SPT schedule for I_P. Then*

$$\sum_{j\in J} C_j^I(\nu) \leq \sum_{j\in J} C_j^{I_P}(\nu_P), \tag{4.6}$$

where $C_j^I(\sigma)$ is the completion time of job j for instance I with schedule σ.

Proof. Suppose (4.6) does not hold. Then $\exists j^* \in J$ such that

$$C_{j^*}^I(\nu) > C_{j^*}^{I_P}(\nu_P). \tag{4.7}$$

Let j^* be the job with smallest index for which this holds. Since none of the machines in I are slower than a machine in I_P, higher completion time directly implies higher starting time. So for j^* we have

$$\begin{aligned} & C_{j^*}^I(\nu) > C_{j^*}^{I_P}(\nu_P) \\ \Leftrightarrow \quad & S_{j^*}^I(\nu) + \frac{p_{j^*}}{s_{\nu_{j^*}}} > S_{j^*}^{I_P}(\nu_P) + p_{j^*} \\ \Rightarrow \quad & S_{j^*}^I(\nu) > S_{j^*}^{I_P}(\nu_P), \end{aligned} \tag{4.8}$$

where $S_j^I(\sigma)$ is the starting time of job j in instance I with schedule σ. (4.8) implies that in schedule ν on each machine there is at least one job with smaller index than j^* with completion time higher than the starting time of j^* in ν_P.

In schedule ν_P, any job with smaller index than j^* has starting time earlier than the starting time of j^*. So there is at most one job on any machine, which has smaller index than j^* and has completion time greater than the starting time of j^*. On the same machine as j^* no job, that has smaller index than j^*, can have completion time greater than the starting time of j^*. So, in schedule ν_P there are at most $m-1$ jobs, that have smaller index than j^*, but have completion time greater than the starting time of j^*, in ν_P. But then, since j^* is the job with smallest index for which (4.7) holds, also in schedule ν there can be at most $m-1$ jobs, with smaller index than j^* with completion time greater than the starting time of j^* in ν_P. So there can not be a job with smallest index for which (4.8) holds. Thus there can not be any such job and (4.6) must hold. □

4.3.4 Upper bounds on the price of anarchy

Combining the resulting bounds for the equilibrium solution and the optimal solution directly gives us the following bounds on the price of anarchy.

From Lemma 3.6 and Lemma 4.4 we get Theorem 4.9.

Theorem 4.9. *Let I be a $Q||\sum C_j$ game instance. Then*

$$\mathrm{POA}(I) \leq \frac{\sum_{i=1}^{m} s_i}{s_m}.$$

When we use Lemma 4.5 together with Lemma 3.6, we get Theorem 4.10.

Theorem 4.10. *Let I be a $Q||\sum C_j$ game instance. Then*

$$\mathrm{POA}(I) \leq \frac{\sum_{j=1}^{n}(n-j+1)p_j}{\sum_{j=1}^{n} p_j}.$$

Combining Lemma's 4.6 and 4.8 results in Theorem 4.11.

Theorem 4.11. *Let I be a $Q||\sum C_j$ game instance. Then*

$$\mathrm{POA}(I) \leq s_m.$$

Taking the result from Lemma 4.7, and applying Lemma 4.4 to (1) of the right hand side of (4.5) and lemma 4.5 to (2) of the right hand side of (4.5), gives us Theorem 4.12.

Theorem 4.12. *Let I be a $Q||\sum C_j$ game instance. Then*

$$\mathrm{POA}(I) \leq 1 + \frac{(m-1)s_m}{\sum_{i=1}^{m} s_i}.$$

Taking the result of Lemma 4.7 and applying 4.4 to both (1) and (2) of the right hand side of (4.5), gives us Theorem 4.13.

Theorem 4.13. *Let I be a $Q||\sum C_j$ game instance. Then*

$$\mathrm{POA}(I) \leq 1 + \frac{(m-1)\sum_{j=1}^{n} p_j}{\sum_{j=1}^{n}(n-j+1)p_j}.$$

For certain specific cases of uniformly related machine scheduling games, the above theorems may give easy constant bounds. Theorem 4.9 and Theorem 4.11 applied to the two machine case result in Theorem 4.14.

Theorem 4.14. *For two machine instances, the price of anarchy is at most* $\frac{1}{2} + \frac{1}{2}\sqrt{5} \approx 1.6180$.

Proof. According to Lemma 3.1, for any instance, we may assume that the slowest machine has speed 1. Let the speed of the fast machine be s. Then, by Theorem 4.9 and Theorem 4.11, the price of anarchy is bounded by $\min\{s, \frac{s+1}{s}\}$. Since s is a strictly increasing function and $\frac{s+1}{s}$ is a strictly decreasing function, the maximum of $\min\{s, \frac{s+1}{s}\}$ is attained at

$$s = \frac{s+1}{s},$$

which is at

$$s = \frac{1}{2} + \frac{1}{2}\sqrt{5} \approx 1.6180.$$

□

The next theorem is our main result. It uses a similar reasoning as explained in section 2.6, where it is used to compute the price of anarchy from (α, β)-smoothness. We also use schedules in which one job j is moved from one machine to another. The difference between our result and a standard smoothness argument is that we need the properties of the optimal solution to prove our bound. Therefore the result does not hold for two arbitrary schedules, but only for an arbitrary schedule and the optimal solution. However the analysis of the robust POA by Roughgarden in [Rou09] is still valid for this slightly less general result, since it does not use that (2.2) holds for any schedule σ^* only for the specific case where σ^* is an optimal schedule. So this result also provides upper bounds on no-regret sequences, course correlated equilibriums and mixed Nash equilibriums.

Theorem 4.15. *The price of anarchy of uniformly related scheduling games with sum of completion times objective and SPT local scheduling rule is no greater than 3.*

Proof. Let μ be the optimal schedule resulting from the MFT algorithm and let ν be an equilibrium schedule. Furthermore let $N_j(i) = |\{j' | \mu_{j'} = i, j' > j\}|$ be the number of jobs with higher index than job j on machine i in μ. Consider the value labels of the machines just before the MFT algorithm placed j on μ_j. The value label of a machine i at that moment is $\frac{N_j(i)+1}{s_i}$. For any two machines i and i' it is clear that

$$\frac{N_j(i)}{s_i} \leq \frac{N_j(i')+1}{s_{i'}} \qquad \forall i, i', \tag{4.9}$$

since either $N_j(i) = 0$ or the left hand side of (4.9) represents the value label of machine i just before the last job with higher index than j was placed on i.

Since that job was not placed on i' (4.9) must hold for any two machines i and i'.

Let $J_i(\sigma)$ be the set of jobs on machine i for any schedule σ. Now consider an arbitrary schedule σ and let (μ_j, σ_{-j}) be the schedule in which every job is scheduled on the machine on which it is scheduled in σ, except for job j which is scheduled on machine μ_j. We compute the completion times for each job j in the corresponding schedule (μ_j, σ_{-j}) and sum over all jobs that are on machine i in μ, so all jobs j such that $\mu_j = i$. We see that

$$\sum_{j \in J_i(\mu)} C_j\left(\mu_j, \sigma_{-j}\right) = \sum_{j \in J_i(\mu)} \left(\frac{p_j}{s_i} + \sum_{\substack{j' \in J_i(\sigma) \\ j' < j}} \frac{p_{j'}}{s_i} \right).$$

Every job j' that is on i in the schedule σ contributes exactly $N_{j'}(i)$ times its processing time on i to this sum, so the above is equal to

$$\begin{aligned}
&= \sum_{j \in J_i(\mu)} \frac{p_j}{s_i} + \sum_{j' \in J_i(\sigma)} N_{j'}(i) \cdot \frac{p_{j'}}{s_i} \\
&\leq \sum_{j \in J_i(\mu)} C_j(\mu) + \sum_{j' \in J_i(\sigma)} \frac{\left(N_{j'}(m) + 1\right) s_i}{s_m} \cdot \frac{p_{j'}}{s_i} \\
&= \sum_{j \in J_i(\mu)} C_j(\mu) + \sum_{j' \in J_i(\sigma)} N_{j'}(m) \cdot \frac{p_{j'}}{s_m} + \sum_{j' \in J_i(\sigma)} \frac{p_{j'}}{s_m} \\
&\leq \sum_{j \in J_i(\mu)} C_j(\mu) + \sum_{j' \in J_i(\sigma)} N_{j'}(m) \cdot \frac{p_{j'}}{s_m} + \sum_{j' \in J_i(\sigma)} C_{j'}(\mu).
\end{aligned}$$

Summing over all machines gives

$$\sum_{j=1}^{n} C_j\left(\mu_j, \sigma_{-j}\right) \leq 2 \sum_{j=1}^{n} C_j(\mu) + \sum_{j=1}^{n} N_j(m) \cdot \frac{p_j}{s_m}.$$

Knowing (4.9), we can bound the last part by

$$\sum_{j=1}^{n} N_j(m) \cdot \frac{p_j}{s_m} \leq \sum_{j=1}^{n} \left(N_j(\mu_j) + 1\right) \cdot \frac{p_j}{s_{\mu_j}} = \sum_{j=1}^{n} C_j(\mu).$$

Since we know that, for an equilibrium solution ν, and any σ_j,

$$C_j\left(\nu_j, \nu_{-j}\right) \leq C_j\left(\sigma_j, \nu_{-j}\right),$$

also

$$\sum_{j=1}^{n} C_j(\nu) \leq \sum_{j=1}^{n} C_j\left(\mu_j, \nu_{-j}\right) \leq 3 \sum_{j=1}^{n} C_j(\mu),$$

for an optimal solution μ. From this we conclude that the price of anarchy is no more than 3. $\square$

Chapter 5

Concluding remarks

In this thesis we proved that 3 is an upper bound on the price of anarchy for unrelated machine scheduling with sum of completion times objective, $Q||\sum C_j$. We also constructed a set of instances which price of anarchy can be arbitrarily close to $\frac{e}{e-1} \approx 1.5820$. For the case in which there are two machines we proved an upper bound of $\frac{1+\sqrt{5}}{2} \approx 1.6180$. We also found an instance on two machines with price of anarchy equal to $\frac{3}{2} \cdot \frac{2+\sqrt{3}}{3+\sqrt{3}} \approx 1.1830$.

These results together with know results from Correa and Queyranne ([CQ10]) establish that the price of anarchy for $Q||\sum C_j$ is less than for $R||\sum w_j C_j$. Comparing with the results from Cole, Gkatzelis and Mirrokni we see that it is possible that for both $Q||\sum C_j$ and $R||\sum C_j$ the price of anarchy is 3, but we believe that for $Q||\sum C_j$ it will turn out to be strictly less than 3.

Since the price of anarchy for $Q||\sum C_j$ is the performance ratio of the SPT algorithm for this scheduling model, our results also apply to that measure.

A few directions of future research are obvious. The first being to prove that the price of anarchy for $Q||\sum C_j$ is strictly less than 3 and, as a logical extension, to close the gap between the lower and the upper bound. Furthermore we haven't looked at the $Q||\sum w_j C_j$ model. It may be interesting to see whether or not the proofs in this thesis, maybe in a slightly modified form, still hold when applied to the more general model with weights.

Appendix A

Used notation

Throughout this thesis we use the following notations unless stated otherwise.

Notation	Denotation
$C_j(\sigma)$	Completion time of job j for some schedule σ
I	Some uniformly related machine scheduling instance
J	The set of jobs
ν	An equilibrium schedule or equilibrium strategy profile
μ	An optimal solution schedule or optimal solution strategy profile
m	The number of machines
M	The set of machines
n	The number of jobs
NE(I)	Highest possible objective value of any equilibrium solution of instance I
OPT(I)	Objective value of the optimal solution of instance I
p_j	Length of job j
POA(I)	The price of anarchy of instance I
$\sigma = (\sigma_j, \sigma_{-j})$	A schedule or strategy profile. Where σ_j is the machine chosen by job j and σ_{-j} is the strategy profile of all jobs but job j
s_i	The speed of machine i

Appendix B

Proofs of analysis steps

Lemma B.1. *If*

$$\frac{A}{B} \leq \frac{C}{D} \leq \frac{C'}{D'} \tag{B.1}$$

and

$$C < C',$$

then

$$\frac{A+C}{B+D} \leq \frac{A+C'}{B+D'}. \tag{B.2}$$

Moreover, if at least one of the inequalities from (B.1) *is strict then* (B.2) *is also strict.*

Proof. From (B.1) we get

$$\begin{aligned}
& \frac{C}{D} \leq \frac{C'}{D'} \\
\Leftrightarrow \quad & \frac{D'}{D} \leq \frac{C'}{C} \\
\Leftrightarrow \quad & \frac{D'}{D} - \frac{D}{D} \leq \frac{C'}{C} - \frac{C}{C} \\
\Leftrightarrow \quad & \frac{D'-D}{D} \leq \frac{C'-C}{C} \\
\Leftrightarrow \quad & \frac{D'-D}{C'-C} \leq \frac{D}{C} \\
\Leftrightarrow \quad & \frac{D'-D}{C'-C} \leq \frac{B}{A} \\
\Leftrightarrow \quad & A(D'-D) \leq B(C'-C).
\end{aligned}$$

All steps are valid since $C' - C > 0$. Now suppose the theorem does not hold, then

$$\begin{aligned}
& \frac{A+C}{B+D} > \frac{A+C'}{B+D'} \\
\Leftrightarrow \quad & (A+C)(B+D') > (A+C')(B+D) \\
\Leftrightarrow \quad & AB + BC + AD' + CD' > AB + BC' + AD + C'D \\
\Rightarrow \quad & BC + AD' > BC' + AD \\
\Leftrightarrow \quad & A(D' - D) > B(C' - C),
\end{aligned}$$

which is a contradiction.

It is easy to see that the whole proof holds for strict inequality if either $\frac{A}{B} < \frac{C}{D}$ or $\frac{C}{D} < \frac{C'}{D'}$. $\square$

The following corollary is an immediate consequence of lemma B.1.

Corollary B.2. *If*

$$\frac{A}{B} \geq \frac{C}{D} \geq \frac{C'}{D'} \tag{B.3}$$

and

$$C > C',$$

then

$$\frac{A+C}{B+D} \leq \frac{A+C'}{B+D'}. \tag{B.4}$$

Moreover, if at least one of the inequalities from (B.3) *is strict then* (B.4) *is also strict.*

Appendix C

$Q||\sum C_j$ is $(2, \frac{1}{2})$-smooth

In [CGM10], Cole, Gkatzelis and Mirrokni prove $(2, 1/2)$-smoothness for unrelated machines with weights and weighted sum of completion times objective. Of course this automatically proves $(2, 1/2)$-smoothness for related machines as well. Here we redo the proof for the $Q||\sum C_j$ model.

Theorem C.1. *SPT local scheduling rule is (2,1/2)-smooth for related machines with sum of completion times objective. And therefore the price of anarchy is no more than 4.*

The following proof follows the proof of Cole, Gkatzelis and Mirrokni [CGM10].

Proof. Let $S_i = \{j|\sigma_j = i\}$ and $S_i^* = \{j|\sigma_j^* = i\}$ be the job sets of machine i for the schedules σ and σ^* respectively. If

$$\sum_{j\in S_i^*} C_j(\sigma_j^*, \sigma_{-j}) \leq \frac{1}{2}\sum_{j\in S_i} C_j(\sigma) + 2\sum_{j\in S_i^*} C_j(\sigma^*) \tag{C.1}$$

holds for every machine i, we are done. To prove (C.1) we first proof that the inequality becomes tighter when all jobs in $S_i \cup S_i^*$ have equal length. Suppose not all jobs in $S_i \cup S_i^*$ have equal length. Then let $\text{Max}_i = \{j \in S_i \cup S_i^* | \forall j' \in S_i \cup S_i^*, p_j \geq p_{j'}\}$ be the set of jobs with maximal length from $S_i \cup S_i^*$. Furthermore, let $J_i = \text{Max}_i \cap S_i$ and $J_i^* = \text{Max}_i \cap S_i^*$ be the maximum length jobs in S_i and S_i^* respectively.

We decrease the length of all jobs in Max_i such that their length is equal to the next longest jobs. This increases the cardinality of Max_i. Let p be this decrease in length. If this decrease in processing time decreases the LHS of (C.1) no more than it decreases the RHS, we can suffice with proving (C.1) for jobs with equal length. So we want to prove:

$$\sum_{j\in J_i^*}\left(\frac{p}{s_i}+\sum_{\substack{j'\in J_i\\ j'<j}}\frac{p}{s_i}\right)\le\frac{1}{2}\sum_{j\in J_i}\sum_{\substack{j'\in J_i\\ j'\le j}}\frac{p}{s_i}+2\sum_{j\in J_i^*}\sum_{\substack{j'\in J_i^*\\ j'\le j}}\frac{p}{s_i}.$$

If we let $A=\sum_{j\in J_i^*}1$ and $B=\sum_{j\in J_i}1$ and on the LHS sum over all $j'\in J_i$ instead of only the $j'<j$, we get

$$\begin{aligned}A+AB&\le\frac{1}{2}\cdot\frac{B^2+B}{2}+2\cdot\frac{A^2+A}{2}\\ 0&\le\frac{B^2}{4}+A^2-AB+\frac{B}{4}\\ 0&\le\left(\frac{B}{2}-A\right)^2+\frac{B}{4},\end{aligned}\tag{C.2}$$

which is true for any $A,B\ge 0$.

Now we may assume that all job lengths are equal. So we assume $p_j=1$ for all j. Then (C.1) becomes:

$$\sum_{j\in S_i^*}\left(1+\sum_{\substack{j'\in S_i\\ j'<j}}1\right)\le\frac{1}{2}\sum_{j\in S_i}\sum_{\substack{j'\in S_i\\ j'\le j}}1+2\sum_{j\in S_i^*}\sum_{\substack{j'\in S_i^*\\ j'\le j}}1.$$

On the LHS we sum over all $j'\in J_i$ again. Then, noticing that $S_i=J_i$ and $S_i^*=J_i^*$, we can rewrite the above as

$$A+AB\le\frac{1}{2}\cdot\frac{B^2+B}{2}+2\cdot\frac{A^2+A}{2},$$

which we already know is true for any $A,B\ge 0$. This proves (C.1). If we now sum over all machines (C.1) becomes

$$\sum_{j=1}^{n}C_j(\sigma_j^*,\sigma_{-j})\le\frac{1}{2}\sum_{j=1}^{n}C_j(\sigma)+2\sum_{j=1}^{n}C_j(\sigma^*),$$

for any two schedules σ and σ^*. This proofs (2,1/2)-smooth for related machines with sum of completion times objective. □

From theorem C.1 it follows that:

Theorem C.2. *The robust price of anarchy of uniformly related scheduling games with sum of completion times objective and SPT local scheduling rule is no greater than 4.*

Bibliography

[CGM10] Richard Cole, Vasilis Gkatzelis, and Vahab Mirrokni. Coordination mechanisms for weighted sum of completion times. *NYU CIMS Technical Report: TR2010-930*, 2010.

[CKN04] G. Christodoulou, E. Koutsoupias, and A. Nanavati. Coordination mechanisms. *Automata, Languages and Programming*, pages 45–56, 2004.

[CQ10] J.R. Correa and M. Queyranne. Efficiency of Equilibria in Restricted Uniform Machine Scheduling with MINSUM Social Cost. *Manuscript*, 2010.

[GLLRK79] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5(2):287–326, 1979.

[HMU07] Birgit Heydenreich, Rudolf Müller, and Marc Uetz. Games and mechanism design in machine scheduling - An introduction. *Production and Operations Management*, 16(4):437–454, 2007.

[HS76] Ellis Horowitz and Sartaj Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *J. Assoc. Comput. Mach.*, 23(2):317–327, 1976.

[IK77] O.H. Ibarra and C.E. Kim. Heuristic algorithms for scheduling independent tasks on nonidentical processors. *Journal of the ACM*, 24(2):289, 1977.

[ILMS09] Nicole Immorlica, Li Li, Vahab S. Mirrokni, and Andreas S. Schulz. Coordination mechanisms for selfish scheduling. *Theoret. Comput. Sci.*, 410(17):1589–1598, 2009.

[Nas50] J.F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America*, 36(1):48–49, 1950.

[Pap01] C. Papadimitriou. Algorithms, games, and the internet. In *Proceedings of the thirty-third annual ACM symposium on Theory of Computing*, pages 749–753. ACM, 2001.

[Pin08] M. Pinedo. *Scheduling: theory, algorithms, and systems.* Springer Verlag, 2008.

[Rou09] Tim Roughgarden. Intrinsic robustness of the price of anarchy. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 513–522, New York, NY, USA, 2009. ACM.

[Smi56] Wayne E. Smith. Various optimizers for single-stage production. *Naval Res. Logist. Quart.*, 3:59–66, 1956.