On the coordination ratio of load balancing problems

Matthias Kretschmer

Rheinische Friedrich-Wilhelm-Universität Bonn
Department of Computer Science V
Römerstr. 164
53117 Bonn

Abstract. We consider the load balancing problem introduced by Koutsoupias and Papadimitriou 1999. Given a set of machines and players where each player own a job, the players can choose the machine that processes their job. The question is how good is a Nash equilibrium in relation to an optimal schedule for a given objective function. The makespan and average completion times are often considered as objective functions. We consider variants where we use the completion time in relation to the job size instead of the completion time and thus normalize by the job sizes. The idea of these objective functions is to model that a player with a small job suffers more from the same amount of delay than a player with a large job. The processing model used by Koutsoupias and Papadimitriou has a bad coordination ratio for such objective functions. With a natural change of the processing model we improve on the performance for the new set of objective functions. In the case of identical machines we give tight upper bounds on the coordination ratio for the unnormalized and normalized objective functions and prove that any Nash equilibrium is an optimal solution for the average and normalized average completion times.

1 Introduction

Load balancing games formalize situations where multiple players want to utilize a set of resources. For example, the resources may be computers used for running jobs and the players have jobs they want to have processed by one of the machines. In contrast to classical problems there exists no central instance that controls to which machine the jobs are assigned. Instead the players choose the machine they utilize.

Of special interest are situations where a player cannot benefit from changing the resource if all other players stick to their chosen resources. Such a state is called a Nash equilibrium and represents a stable state of the game. Given an objective function one is able to measure the performance of the decentralized game by comparing the worst case stable solution (the worst case Nash equilibrium) to the optimal solution with respect to the objective function.

One has to carefully select the objective function as it models the aspect of the game which should be optimized. In preceding works the absolute maximum completion time and average completion times of the jobs are used as the objective function.

In this paper we use another form of an objective function. The motivation for the use of other objective functions is as follows: the completion time should be measured in relation to the job size, as the same amount of delay until a job is completed is worse for a player with a small job in comparison to a player with a large job. Players with large jobs expect that the job will take some time for processing, while one wants to have a small job processed immediately. This is modeled by the normalized maximum and average completion time, which is the maximum and average of all completion times in relation to the job sizes. N. Blum [2] raised the question how to define a model that has a good performance for normalized objective functions. We present a model, which guarantees good performance for the normalized and unnormalized objective functions.

In Section 2 we give a formal introduction on the used framework. Koutsoupias and Papadimitriou [13] introduced the KP-model. In the KP-model each resource processes all the jobs before returning the answer to the player. This emposes a very poor performance in respect to the normalized objective functions. In Section 3 we give a formal introduction to the KP-model, present known results of the performance of the KP-model for unnormalized objective functions and show how poor it performs with respect to the normalized objective functions. Section 4 introduces the priority model and shows that the performance is much better than the performance of the KP-model for most of the considered objective functions.

2 Basics and Notation

Koutsoupias and Papadimitriou [13] introduced load balancing games as strategic games – or games in normal form [3,4,15,16]. Instead of giving an introduction to the generic framework, we give a specialization of the model as load balancing games.

Let $G = \langle N, M, (x_j), (s_i), C\rangle$ be a *load balancing game* where

- $N := \{1, \ldots, n\}$ is the non-empty *set of players* (we will use the term job synonymously for player),
- $M := \{1, \ldots, m\}$ is the non-empty *set of machines*,
- for all $j \in N$: $x_j \in \mathbb{R}_+$ is the *size of the job of player* j,
- for all $i \in M$: $s_i \in \mathbb{R}_+$ is the *speed of machine* i and
- $C : N \times M^N \to \mathbb{R}$ with $C_j(a_1, \ldots, a_n) := C(j, a_1, \ldots, a_n)$ is the *completion time of the job of player* j when players $k \in N$ choose machine $a_k \in M$.

$a_j \in M$ is called a (*pure*) *strategy for player* j and $A = (a_1, \ldots, a_n)$ is called a (*pure*) *strategy profile* or (*pure*) *profile*. The players choose their strategy once

and cannot change it later. Load balancing games are non-cooperative and uncoordinated strategic games, that means, players choose their strategy without knowledge of the strategies of the other players and without communicating with the other players.

If all machines are of equal speed, we also write $\langle N, M, (x_j), C\rangle$ instead of $\langle N, M, (x_j), (1), C\rangle$. The speeds of the machines are linear factors in the models we consider. This allows us to normalize the games where all machines have equal speed to games where the speeds of all machines are 1 by scaling the jobs. When considering games with machines of equal speed we always assume that the machines' speed is 1.

An example for such a model is the *KP-Model* which was first presented by Koutsoupias and Papadimitriou [13]. The authors defined the completion time of job j to be the running time of the machine that processes this job. The *completion time in the KP-Model* C_j^{KP} is

$$C_j^{\mathrm{KP}}(a_1, \ldots, a_n) := \frac{1}{s_{a_j}} \sum_{k:a_k=a_j} x_k.$$

We want to look at the drawbacks of decentralized egoistic machine selection. The Nash equilibrium that we will introduce defines a stable state of the game where a player – if it would be allowed – would not change his strategy, because it would not reduce the completion time. We want to compare the Nash equilibria of a game to the optimal solution for an objective function.

A (*pure*) *Nash equilibrium* of a game $G = \langle N, M, (x_j), (s_i), C\rangle$ is a profile $A = (a_1, \ldots, a_n)$ such that for all players $j \in N$ and all machines $i \in M$

$$C_j(a_1, \ldots, a_n) \leq C_j(a_1, \ldots, a_{j-1}, i, a_{j+1}, \ldots, a_n).$$

The *social optimum* $\mathrm{OPT}(f, G)$ *of the objective function* f *and game* $G = \langle N, M, (x_j), (s_i), C\rangle$ is the social value of a profile that minimizes f.

$$\mathrm{OPT}(f, G) = \min\{f(G, A) \mid A \text{ is a profile of } G\}$$

The *coordination ratio* $\mathrm{CR}(f, G)$ is a measure of the quality of a Nash equilibrium. We measure this by the ratio of a Nash equilibrium that maximizes the objective function f to the social optimum. The social value of such a Nash equilibrium is called *worst-case Nash equilibrium* $\mathrm{WN}(f, G)$ *of game* G *to* f where

$$\mathrm{WN}(f, G) = \max\{f(G, A) \mid A \text{ is a Nash equilibrium of } G\}.$$

The coordination ratio $\mathrm{CR}(f, G)$ of game G for the objective function f is

$$\mathrm{CR}(f, G) := \frac{\mathrm{WN}(f, G)}{\mathrm{OPT}(f, G)}.$$

Objective functions of special interest are the maximum completion time

$$C_{\max}(G, A) := \max_{j \in N} C_j(A)$$

and the average completion time

$$\Sigma C_j(G, A) := \sum_{j \in N} C_j(A).$$

Both are found in literature that considers load balancing games and scheduling in general. For the average completion time we use the sum of the completion times instead of the mean as the number of players is just a constant factor and in the case of the coordination ratio it would vanish in the fraction.

If we consider load balancing games we may want to measure the performance from the point of view of players. We want to model that players register the completion time relative to the sizes of their jobs. To do this we introduce the normalized maximum completion time $C^n_{\max}$ and normalized average completion time ΣC^n_j:

$$C^n_{\max}(G, A) := \max_{j \in N} \frac{C_j(A)}{x_j} \text{ and}$$

$$\Sigma C^n_j(G, A) := \sum_{j \in N} \frac{C_j(A)}{x_j}.$$

3 The KP-Model

In this section the KP-model will be considered, i.e. the completion time of a job j is defined as

$$C_j(A) := C^{\mathrm{KP}}_j(A) = \sum_{\substack{k \in N \\ a_k = a_j}} x_k.$$

This means that a machine process all jobs before returning the result to the players. Thus a job of a player is delayed by any job processed on that machine.

3.1 Maximum Completion Time and Average Completion Time

We may wonder if there is a constant upper bound of the coordination ratio for the maximum completion time $C_{\max}$. Gairing et al. [10] give an answer to this question by providing a tight upper bound for identical machines.

Theorem 1. *[10, Theorem 9] Let $G = \langle N, M, (x_j), C^{\mathrm{KP}} \rangle$ be a load balancing game. Then*

$$\mathrm{CR}(C_{\max}, G) \leq 2 - \frac{2}{m+1}$$

and this bound is tight.

In the case of nonidentical machines there is no constant upper bound, but Feldmann et al. [7] provide an asymptotically tight upper bound depending on the number of machines. Let Γ be the Gamma function.

Theorem 2. *[7, Theorem 3] Let $G = \langle N, M, (x_j), (s_i), C^{\mathrm{KP}} \rangle$ be a load balancing game with $m = |M|$ machines. Then*

$$\mathrm{CR}(C_{\max}, G) \leq \Gamma^{-1}(m)$$

and this bound is asymptotically tight.

In contrast to the maximum completion time there is no constant upper bound for the average completion time for identical machines.

Berenbrink et al. [1] provide a worst-case lower bound in the number of players for the avarage completion time.

Theorem 3. *[1, Lemma 3.2] For all number of players $n \in \mathbb{N}$ there exists a load balancing game $G = \langle N, M, (x_j), C^{\mathrm{KP}} \rangle$ such that*

$$\mathrm{CR}(\Sigma C_j, G) \geq \frac{n}{5}.$$

Not all load balancing games with n players have this high coordination ratio, but there exists some that have. The upper bound given by Berenbrink et al. shows that the coordination ratio is bounded from above by the maximum job size $x_{\max} := \max\{x_k \mid k \in N\}$.

Theorem 4. *[1, Theorem 2.5] For all load balancing games $G = \langle N, M, (x_j), C^{\mathrm{KP}} \rangle$*

$$\mathrm{CR}(\Sigma C_j, G) \leq 4x_{\max}.$$

Feldmann et al. [8] give an overview of the results up to 2003 about load balancing games in the KP-model. They cover results related to pure strategies and mixed strategies (mixed strategies are random distributions on the actions). Further specific results can be found in [1,6,7,9,10,13].

3.2 Normalized Objective Functions

In the case of the normalized objective functions, we will see that the coordination ratio is unbounded, i.e. for every $\gamma > 1$ there is a game $G = \langle N, M, (x_j), C^{\mathrm{KP}} \rangle$ such that

$$\mathrm{CR}(C^n_{\max}, G) \geq \gamma \text{ and } \mathrm{CR}(\Sigma C^n_j, G) \geq \gamma. \tag{1}$$

For example consider the following game with four players and two machines:

- $N := \{1, 2, 3, 4\}$
- $M := \{1, 2\}$

– $x_1 := x_2 := 1$, $x_3 := x_4 := x$ where $x > 3$

In both cases (ΣC_j^n and $C_{\max}^n$) the optimal solution is to place the jobs of the same size on the same machine. This implies that $B = (1, 1, 2, 2)$ is a social optimum. A Nash equilibrium may be generated by the LPT list-scheduling rule [9, Theorem 2] giving the Nash equilibrium A with

$$A = (1, 2, 1, 2).$$

The values of the objective functions ΣC_j^n and $C_{\max}^n$ for both profiles are:

$$\Sigma C_j^n(B) = 2\frac{2 \cdot 1}{1} + 2\frac{2x}{x} = 8,$$

$$\Sigma C_j^n(A) = 2\frac{1+x}{1} + 2\frac{1+x}{x} \geq 2(1+x),$$

$$C_{\max}^n(B) = \max\left\{\frac{2 \cdot 1}{1}, \frac{2x}{x}\right\} = 2 \text{ and}$$

$$C_{\max}^n(A) = \max\left\{\frac{1+x}{1}, \frac{1+x}{x}\right\} = 1 + x.$$

Hence the coordination ratios of both normalized objective functions are

$$\mathrm{CR}(\Sigma C_j^n, G) \leq \frac{2(1+x)}{8} = \frac{1+x}{4} \text{ and}$$

$$\mathrm{CR}(C_{\max}^n, G) = \frac{1+x}{2}.$$

Choosing $x > 3$ such that $\frac{1+x}{4} > \gamma$ implies (1). Thus there is no constant upper bound, even if the number of players and machines is constant.

4 The Priority Model

We introduce the priority model to improve the situation for normalized objective functions. We will even show that the coordination ratio for the average completion time in the case of identical machines improves.

We require that the sizes of the jobs are known by the machines before running the jobs. The jobs will be processed on the machines in ascending order of the job sizes. The completion time of a job is the time when the machine finishes processing it (instead of the processing time of all jobs on that machine). This ordering ensures that small jobs will not be delayed by large jobs which should improve the normalized coordination ratios.

To define the completion time we have to give a total ordering of the jobs. We want to process jobs of small sizes first, so small jobs have a higher priority than large jobs. The ordering of jobs of the same size is not clear. We just use any

order which has to be given before the game starts, i.e. the order is included in the game rules. We give the order by the priority relation $\preceq$ which defines a total ordering of all jobs such that

$$\forall j,k \in N : x_j < x_k \Rightarrow j \preceq k.$$

We can now define the *completion time of the priority model* C_j^P for a profile $A = (a_1, \ldots, a_n)$ by

$$C_j^P(A) := \sum_{\substack{k:a_k=a_j \\ k \preceq j}} \frac{x_k}{s_{a_j}}.$$

We will number the jobs such that $1 \preceq 2 \preceq \cdots \preceq n$.

4.1 Maximum Completion Time and Average Completion Time

Intrestingly the List Scheduling algorithm [11] obeying the order given by $\preceq$ provides an algorithm that generates a Nash equilibrium for the priority model. The List Scheduling algorithm is a greedy algorithm for allocating jobs to machines. In each step the algorithm selects an unassigned job and places it on a machine that would finish first the processing of the job after processing the jobs already assigned to this machine. In our case the jobs are assigned in the order given by $\preceq$ which is equivalent to the SPT-rule. Algorithm $\mathrm{LS}_{\preceq}$ formalizes the algorithm. Immorlica et al. [12] have shown that this algorithm generates a Nash equilibrium and that any Nash equilibrium in the priority model may be generated by the List Scheduling algorithm obeying the order given by $\preceq$. This allows us to use the results of the performance of List Scheduling as an approximation algorithm for minimizing the maximum completion time and the average completion time to derive the coordination ratio for $C_{\max}$ and ΣC_j.

Algorithm 1. $\mathrm{LS}_{\preceq}$

Input: load balancing game $\langle N, M, (x_j), (s_i), C^P \rangle$, priority relation $\preceq$; the jobs are ordered such that $1 \preceq 2 \preceq \cdots \preceq n$
Output: profile $A = (a_1, \ldots, a_n)$

1. $A := (a_1, \ldots, a_n)$, $\forall j \in N : a_j := 0$ (at the beginning each job is processed by no machine)
2. **for** $j := 1$ **to** n **do**
   (a) select $i \in M$, such that $\forall l \neq i : C_j^P(A_{-j}, i) \leq C_j^P(A_{-j}, l)$
   (b) $a_j := i$
3. **return** A

List Scheduling with the order defined by $\preceq$ is a special variant of the SPT-rule. The SPT-rule for List Scheduling defines that the jobs are scheduled in non-decreasing order. It is known that the List Scheduling algorithm with the SPT-rule is an optimal algorithm for the avarage completion time [5]. List Scheduling with the SPT-rule is a $2 - \frac{1}{m-1}$ approximation algorithm for the maximum completion time [11]. This bound is tight. Hence, we get the following corollary.

Corollary 1. *For all load balancing games* $G = \langle N, M, (x_j), C^P \rangle$ *with* $m = |M|$ *machines*

$$\mathrm{CR}(C_{\max}, G) \leq 2 - \frac{1}{m} \qquad \textit{and} \qquad \mathrm{CR}(\Sigma C_j, G) = 1.$$

This result is an improvement for the objective function ΣC_j compared to the KP-model. In the case of $C_{\max}$ the result is slightly worse than the upper bound of $2 - \frac{2}{m+1}$ of the KP-model.
Liu and Liu [14] showed an upper bound for the approximation ratio of the List Scheduling algorithm for arbitrary machines. This gives the following corollary:

Corollary 2. *Let* $G = \langle N, M, (x_j), (s_i), C^P \rangle$ *be a load balancing game, then*

$$\mathrm{CR}(C_{\max}, G) \leq 1 + \frac{\max_{i \in M} s_i}{\min_{l \in M} s_l} + \frac{\max_{i \in M} s_i}{\sum_{l \in M} s_l}.$$

In the case of arbitrary machines and the average completion time, we can prove a similar upper bound.

Lemma 1. *Let* $G = \langle N, M, (x_j), (s_i), C^P \rangle$ *be a load balancing game, then*

$$\mathrm{CR}(\Sigma C_j, G) \leq \frac{\max_{i \in M} s_i}{\min_{l \in M} s_l} + \frac{\max_{i \in M} s_i}{\sum_{l \in M} s_l} n.$$

Proof. Let $L_j^i(A)$ the unnormalized load of machine i considering only the players $k \leq j$, i.e.,

$$L_j^i(A) = \sum_{\substack{k:k \leq j \\ a_k = i}} \frac{x_k}{s_i}.$$

Then for all players $j \in N$ and all profiles A

$$C_j(A) = L_j^{a_j}(A).$$

Let $A = (a_1, \ldots, a_n)$ be a Nash equilibrium, such that the $\mathrm{WN}(\Sigma C_j, G) = \Sigma C_j(A)$. Then

$$\mathrm{WN}(\Sigma C_j, G) = \Sigma C_j(A) = \sum_{j \in N} C_j(A) = \sum_{j \in N} L_j^{a_j}(A).$$

Let for all players $j \in N$ the machine l_j be the machine with the lowest load considering only the players $\leq j$, i.e., $\forall i \in M : L_j^{l_j}(A) \leq L_j^i(A)$. Then

$$C_j(A) = L_j^{a_j}(A) \leq L_j^{l_j}(A) + \frac{x_j}{s_{l_j}}.$$

Otherwise A would not be a Nash equilibrium, because player j could move to machine l_j to decrease the completion time. All players $k > j$ do not influence the

completion time of player j. Hence, the completion time of player j on machine l_j is

$$L_j^{l_j}(A) + \frac{x_j}{s_{l_j}}.$$

This implies

$$\mathrm{WN}(\Sigma C_j, G) = \sum_{j \in N} L_j^{a_j}(A) \leq \sum_{j \in N} \left(L_j^{l_j}(A) + \frac{x_j}{s_{l_j}} \right). \tag{2}$$

Now we give an upper bound for $L_j^{l_j}(A)$. Consider the following sum

$$\sum_{i \in M} s_i L_j^i(A).$$

By definition $L_j^i(A) \geq L_j^{l_j}(A)$ for all $i \in M$. Hence

$$\sum_{i \in M} s_i L_j^i(A) \geq \sum_{i \in M} s_i L_j^{l_j}(A) = L_j^{l_j}(A) \sum_{i \in M} s_i.$$

On the other hand

$$\sum_{i \in M} s_i L_j^i(A) = \sum_{i \in M} s_i \left(\sum_{\substack{k : k \leq j \\ a_k = i}} \frac{x_k}{s_i} \right) = \sum_{k \leq j} x_k.$$

Both terms give the following upper bound for $L_j^{l_j}(A)$:

$$L_j^{l_j}(A) \leq \frac{\sum_{k \leq j} x_k}{\sum_{i \in M} s_i}.$$

After insertion of this upper bound into (2) we obtain

$$\begin{aligned} \mathrm{WN}(\Sigma C_j, G) &\leq \sum_{j \in N} \left(L_j^{l_j}(A) + \frac{x_j}{s_{l_j}} \right) \\ &\leq \sum_{j \in N} \left(\frac{\sum_{k \leq j} x_k}{\sum_{i \in M} s_i} + \frac{x_j}{s_{l_j}} \right) \\ &\leq n \frac{\sum_{j \in N} x_j}{\sum_{i \in M} s_i} + \frac{\sum_{j \in N} x_j}{\min_{i \in M} s_i}. \end{aligned}$$

We now give a lower bound for a social optimum to obtain the coordination ratio. Let $B = (b_1, \ldots, b_n)$ be a profile that optimizes the objective function ΣC_j. Each job $j \in N$ has to be processed on some machine $i \in M$. On that machine at least te jbo itself is scheduled. A simple lower bound is thus

$$\Sigma C_j(B) = \sum_{j \in N} C_j(B) \geq \sum_{j \in N} \frac{x_j}{s_{b_j}} \geq \sum_{j \in N} \frac{x_j}{\max_{i \in M} s_i}.$$

Combining the upper bound on A with the lower bound on B gives

$$\begin{aligned}\mathrm{WN}(\Sigma C_j, G) &\leq n\frac{\sum_{j\in N} x_j}{\sum_{i\in M} s_i} + \frac{\sum_{j\in N} x_j}{\min_{i\in M} s_i} \\ &= n\frac{\sum_{j\in N} x_j}{\max_{i\in M} s_i}\frac{\max_{i\in M} s_i}{\sum_{i\in M} s_i} + \frac{\sum_{j\in N} x_j}{\max_{i\in M} s_i}\frac{\max_{i\in M} s_i}{\min_{i\in M} s_i} \\ &\leq \left(n\frac{\max_{i\in M} s_i}{\sum_{i\in M} s_i} + \frac{\max_{i\in M} s_i}{\min_{i\in M} s_i}\right)\mathrm{OPT}(\Sigma C_j, G).\end{aligned}$$

□

4.2 Normalized Objective Functions

In this section we will consider normalized objective functions. We have chosen the priority model to overcome the poor performance of the KP-model for those objective functions. We would expect, that the priority model behaves much better and in fact we can prove that the coordination ratio for the normalized objective functions is the same as for the absolute objective functions in the case of identical machines.

Theorem 5. *Let $G = \langle N, M, (x_j), C^P\rangle$ be a load balancing game with $m = |M|$ machines. Then*

$$\mathrm{CR}(C^n_{\max}, G) \leq 2 - \frac{1}{m}.$$

The proof of Theorem 5 is similiar to the proof that List Scheduling is a $2 - \frac{1}{m}$ approximation algorithm.

This shows that we can reproduce the results of the unnormalized case $C_{\max}$ for the normalized case $C^n_{\max}$. This is a vast improvement to the KP-model where the coordination ratio is not bounded by a constant or the number of players and machines. We now want to show, that similar results may be reproduced for the case of the objective function ΣC^n_j. We can prove that all Nash equilibria have the same social value as the social optimum.

Theorem 6. *Let $G = \langle N, M, (x_j), C^P\rangle$ be a load balancing game in the priority model, then*

$$\mathrm{CR}(\Sigma C^n_j, G) = 1.$$

Proof. The proof will be performed in two steps. At first we show that all Nash equilibria have the same social cost (Lemma 2). Then we show that there exists a Nash equilibrium with the same social cost as a social optimum (Lemma 3). The combination of both lemmas proves the theorem.

Lemma 2. *Let $G = \langle N, M, (x_j), C^P\rangle$ be a load balancing game. Then for all Nash equilibria A and B of G*

$$\Sigma C^n_j(G, A) = \Sigma C^n_j(G, B).$$

Proof. We prove the lemma by induction on the number of players. Let $A = (a_1, \ldots, a_n)$ and $B = (b_1, \ldots, b_n)$ be two Nash equilibria with n players of the same load balancing game $G = \langle N, M, (x_j), C^P \rangle$.
Instead of proving that the social cost is equal for both Nash equilibria, we show that for each player $1 \leq j \leq n$ the completion time is the same:

$$C_j^P(A) = C_j^P(B).$$

We do this by proving that there is a permutation $\pi : M \to M$ on the machines, such that for each machine $1 \leq i \leq m$ the load

$$L^i(C) = \sum_{\substack{k:k\leq n \\ a_k = i}} x_k$$

is equal in both Nash equilibria: $L^i(A) = L^{\pi(i)}(B)$.

- $n = 1$: All machines have the same speed so the completion time is the same on all machines. Hence there exists such a permutation and $C_1^P(A) = C_1^P(B)$.
- $n - 1 \to n$: $A' = (a_1, \ldots, a_{n-1})$ and $B' = (b_1, \ldots, b_{n-1})$ are both Nash equilibria, because job n is the lowest priority job and thus is always scheduled at last.
  By the induction hypothesis for all jobs $j : 1 \leq j \leq n-1$ the completion times for both profiles are the same

  $$C_j^P(A') = C_j^P(B')$$

  and there exists a permutation π such that for all machines $i : 1 \leq i \leq m$

  $$L^i(A') = L^{\pi(i)}(B').$$

  All Nash equilibria are generated by the List Scheduling algorithm obeying the order implied by $\preceq$ and all machines are of equal speed. Hence we put job n on a machine with the lowest load and $L^{a_n}(A') = L^{b_n}(B')$. Thus $L^{a_n}(A) = L^{a_n}(A') + x_n = L^{b_n}(B') + x_n = L^{b_n}(B)$. Now we create the permutation π' such that

  $$\pi'(i) = \begin{cases} b_n, & \text{if } i = a_n \\ a_n, & \text{if } i = b_n \\ \pi(a_n), & \text{if } a_n = \pi(i) \\ \pi(b_n), & \text{if } b_n = \pi(i) \\ \pi(i), & \text{otherwise.} \end{cases}$$

  Permutation π' ensures that for all machines $i : 1 \leq i \leq m$ the load is equal: $L^i(A) = L^{\pi(i)}(B)$.
  Adding job n to the profiles A' and B' does not change the completion time of the jobs $j \leq n-1$ and the completion time of job n is $C_n^P(A) = \frac{L^{a_n}(A)}{x_n} = \frac{L^{b_n}(B)}{x_n} = C_n^P(B)$. Hence the completion times of all jobs and the social costs are equal. □

Lemma 3. *Let $G = \langle N, M, (x_j), C^P \rangle$ be a load balancing game. Then there exists a Nash equilibrium A, such that*

$$\Sigma C_j^n(G, A) = \mathrm{OPT}(\Sigma C_j^n, G).$$

Proof. To prove the lemma, we construct an algorithm that transforms an arbitrary optimal profile to a Nash equilibrium which does not have a higher social value. We do this by specifying an algorithm that does the transformation and ensures that in each step the social cost will not increase.

Similiar to the List Scheduling algorithm we use a greedy scheme. Starting by the highest priority job, we move each job to a new machine if a job may decrease its completion time by changing its strategy. In such a case we choose the machine which minimizes the completion time. This is algorithm $\mathcal{A}_1$ and will be used as a basis to create an algorithm that will fit our needs.

Algorithm $\mathcal{A}_1$.
Input load balancing game $G = \langle N, M, (x_j), C^P \rangle$, social optimum $A^0 = (a_1^0, \ldots, a_n^0)$
Output Nash equilibrium $A^n = (a_1^n, \ldots, a_n^n)$.

1. **for** $j := 1$ **to** n **do**
   - $A^j := A^{j-1}$
   - **if** $\exists i \in M : C_j^P(A_{-j}^j, i) < C_j^P(A^j)$ **then**
     - $a_j^j := \arg\min_{i \in M} C_j^P(A_{-j}^j, i)$
2. **return** A^n

For the very same reasons why List Scheduling obeying the order given by $\preceq$ generates a Nash equilibrium, this algorithm transforms a profile to a Nash equilibrium (the completion time of a touched job will not change, because afterwards only lower priority jobs are moved).

The problem with this algorithm is, that the social cost may increase after a step. Let j be the first job where the completion time increases. In this step the social cost changes, so we move the job j from machine a_j^{j-1} to machine a_j^j. Consider the social cost of profile A^j which is

$$\Sigma C_j^n(A^j) = \Sigma C_j^n(A^{j-1}) + \frac{C_j^P(A^j) - C_j^P(A^{j-1})}{x_j} + \sum_{\substack{k:k>j \\ a_k^{j-1} = a_j^j}} \frac{x_j}{x_k} - \sum_{\substack{k:k>j \\ a_k^j - 1 = a_j^{j-1}}} \frac{x_j}{x_k}.$$

We compensate this change by moving the job on machine a_j^j which is processed directly after job j to machine a_j^{j-1}. Figure 1 visualizes this change. The new algorithm is given below.

Algorithm $\mathcal{A}_2$.
Input load balancing game $G = \langle N, M, (x_j), C^P \rangle$, social optimum $A^0 = (a_1^0, \ldots, a_n^0)$
Output Nash equilibrium $A^n = (a_1^n, \ldots, a_n^n)$.

1. **for** $j := 1$ **to** n **do**

- $A^j := A^{j-1}$
- **if** $\exists i \in M : C_j^P(A^j_{-j}, i) < C_j^P(A^j)$ **then**
  - $a_j^j := \arg\min_{i \in M} C_j^P(A^j_{-j}, i)$
- $A'^j := A^j$
- **if** $\Sigma C_j^n(A^j) > \Sigma C_j^n(A^{j-1})$ **then**
  - $\overline{j} := \min\{k > j \mid a_k^{j-1} = a_j^j\}$
  - $a_{\overline{j}}^j := a_j^{j-1}$

2. **return** A^n

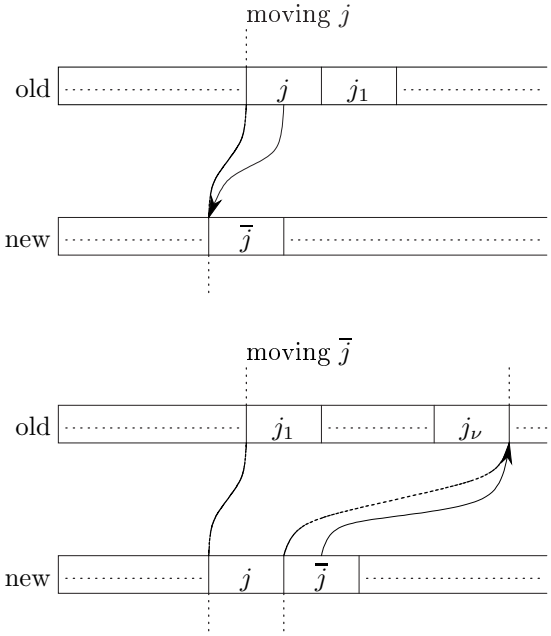

Fig. 1. Modifition of $\mathcal{A}_1$

Let $\overline{j} := \min\{k > j \mid a_k^{j-1} = a_j^j\}$ be the job that is processed directly after job j on machine a_j^j. This job has a lower priority than job j. So after moving j we will touch this job again and hence the result of algorithm $\mathcal{A}_2$ will be a Nash equilibrium.

The problem with this modification is, that there might be jobs $j_1, \ldots, j_\nu$ on machine a_j^{j-1} with a lower priority than j and higher priority than $\overline{j}$. Let us

consider the new social cost for profile A^j:

$$\begin{aligned}\Sigma C_j^n(A^j) &= \Sigma C_j^n(A'^j) + \frac{C_{\overline{j}}^P(A^j) - C_{\overline{j}}^P(A'^j)}{x_{\overline{j}}} + \sum_{\substack{k:k>\overline{j}\\ a_k^{j-1}=a_j^{j-1}}} \frac{x_{\overline{j}}}{x_k} - \sum_{\substack{k:k>\overline{j}\\ a_k^{j-1}=a_j^{j}}} \frac{x_{\overline{j}}}{x_k} \\ &= \Sigma C_j^n(A^{j-1}) + \frac{C_{\overline{j}}^P(A^j) - C_{\overline{j}}^P(A'^j)}{x_{\overline{j}}} + \sum_{\substack{k:k>\overline{j}\\ a_k^{j-1}=a_j^{j-1}}} \frac{x_{\overline{j}}}{x_k} - \sum_{\substack{k:k>\overline{j}\\ a_k^{j-1}=a_j^{j}}} \frac{x_{\overline{j}}}{x_k} \\ &\quad + \frac{C_j^P(A'^j) - C_j^P(A^{j-1})}{x_j} + \sum_{\substack{k:k>j\\ a_k^{j-1}=a_j^{j-1}}} \frac{x_j}{x_k} - \sum_{\substack{k:k>j\\ a_k^{j-1}=a_j^{j}}} \frac{x_j}{x_k}\end{aligned}$$

We simplify the equation and start by expressing $C_{\overline{j}}^P$ in terms of C_j^P:

- $C_{\overline{j}}^P(A'^j) = C_j^P(A'^j) + x_{\overline{j}}$ (follows from the definition of $\overline{j}$)
- Let $J = \{j_1, \ldots, j_\nu\} = \{k \mid j < k < \overline{j} \wedge a_k^{j-1} = a_j^{j-1}\}$. Then

$$C_{\overline{j}}^P(A^j) = C_j^P(A^{j-1}) - x_j + \sum_{k\in J} x_k.$$

This gives us

$$\begin{aligned}\Sigma C_j^n(A^j) &= \Sigma C_j^n(A^{j-1}) + \frac{C_j^P(A^{j-1}) - C_j^P(A'^j)}{x_{\overline{j}}} + \frac{C_j^P(A'^j) - C_j^P(A^{j-1})}{x_j} \\ &\quad + \sum_{k\in J} \frac{x_k}{x_{\overline{j}}} - \frac{x_j}{x_{\overline{j}}} + x_{\overline{j}} \left(\sum_{\substack{k:k>\overline{j}\\ a_k^{j-1}=a_j^{j-1}}} \frac{1}{x_k} - \sum_{\substack{k:k>\overline{j}\\ a_k^{j-1}=a_j^{j}}} \frac{1}{x_k} \right) \\ &\quad + x_j \left(\sum_{\substack{k:k>j\\ a_k^{j-1}=a_j^{j}}} \frac{1}{x_k} - \sum_{\substack{k:k>j\\ a_k^{j-1}=a_j^{j-1}}} \frac{1}{x_k} \right) \\ &\leq \Sigma C_j^n(A^{j-1}) + \sum_{k\in J} \frac{x_k}{x_{\overline{j}}} - \frac{x_j}{x_{\overline{j}}} + x_{\overline{j}} \left(\sum_{\substack{k:k>\overline{j}\\ a_k^{j-1}=a_j^{j-1}}} \frac{1}{x_k} - \sum_{\substack{k:k>\overline{j}\\ a_k^{j-1}=a_j^{j}}} \frac{1}{x_k} \right) \\ &\quad + x_j \left(\sum_{\substack{k:k>j\\ a_k^{j-1}=a_j^{j}}} \frac{1}{x_k} - \sum_{\substack{k:k>j\\ a_k^{j-1}=a_j^{j-1}}} \frac{1}{x_k} \right)\end{aligned}$$

Now we split up the last four sums and give an upper bound where these terms are eliminated. This reduces the equation to

$$\Sigma C_j^n(A^j) \leq \Sigma C_j^n(A^{j-1}) + \leq \sum_{k\in J} \frac{x_k}{x_{\overline{j}}} - \frac{x_j}{x_{\overline{j}}} - x_{\overline{j}} \left(\sum_{\substack{k:k>\overline{j} \\ a_k^{j-1}=a_j^j}} \frac{1}{x_k} - \sum_{\substack{k:k>j \\ a_k^{j-1}=a_j^{j-1}}} \frac{1}{x_k} \right)$$

$$- \sum_{k\in J} \frac{x_{\overline{j}}}{x_k} + x_j \left(\sum_{\substack{k:k>\overline{j} \\ a_k^{j-1}=a_j^j}} \frac{1}{x_k} - \sum_{\substack{k:k>j \\ a_k^{j-1}=a_j^{j-1}}} \frac{1}{x_k} \right) + \frac{x_j}{x_{\overline{j}}}$$

$$= \sum_{k\in J} \left(\frac{x_k}{x_{\overline{j}}} - \frac{x_{\overline{j}}}{x_k} \right) + (x_j - x_{\overline{j}}) \left(\sum_{\substack{k:k>\overline{j} \\ a_k^{j-1}=a_j^j}} \frac{1}{x_k} - \sum_{\substack{k:k>j \\ a_k^{j-1}=a_j^{j-1}}} \frac{1}{x_k} \right).$$

Let us now consider the first sum. By the definition of J it follows that for all $k \in J : x_k \leq x_{\overline{j}}$. Hence $\forall k \in J : \frac{x_k}{x_{\overline{j}}} \leq 1 \leq \frac{x_{\overline{j}}}{x_k}$ and

$$\sum_{k\in J} \left(\frac{x_k}{x_{\overline{j}}} - \frac{x_{\overline{j}}}{x_k} \right) \leq 0.$$

This reduces our equation to

$$\Sigma C_j^n(A^j) \leq \Sigma C_j^n(A^{j-1}) + (x_j - x_{\overline{j}}) \left(\sum_{\substack{k:k>\overline{j} \\ a_k^{j-1}=a_j^j}} \frac{1}{x_k} - \sum_{\substack{k:k>j \\ a_k^{j-1}=a_j^{j-1}}} \frac{1}{x_k} \right).$$

From the definition of $\overline{j}$ it follows that $x_j - x_{\overline{j}} \leq 0$. So we need to prove that

$$\sum_{\substack{k:k>\overline{j} \\ a_k^{j-1}=a_j^j}} \frac{1}{x_k} - \sum_{\substack{k:k>j \\ a_k^{j-1}=a_j^{j-1}}} \frac{1}{x_k} \geq 0. \tag{3}$$

We do this by considering the contradiction and generating a profile $B = (b_1, \ldots, b_n)$ which will then have a social cost less than the optimum A^0.

Profile B is equal to profile A'^j except that all jobs with a priority less than j from machine a_j^{j-1} are moved to machine a_j^j and all jobs with a priority less than j from machine a_j^j are moved to machine a_j^{j-1}. Formally the profile B is given by

$$b_k := \begin{cases} a_j^j, & \text{if } k \geq j \wedge a_k^{j-1} = a_j^{j-1} \\ a_j^{j-1}, & \text{if } k \geq j \wedge a_k^{j-1} = a_j^j \\ a_k^{j-1}, & \text{otherwise.} \end{cases}$$

We now calculate the social cost of profile B. Let $\delta := C_j^P(B) - C_j^P(A^{j-1})$ the change of job j. From the definition of profile j it follows that $\delta < 0$. Hence

$$\Sigma C_j^n(B) = \Sigma C_j^n(A^{j-1}) + \delta \left(\sum_{\substack{k:k \geq j \\ a_k^{j-1} = a_j^{j-1}}} \frac{1}{x_k} - \sum_{\substack{k:k \geq j \\ a_k^{j-1} = a_j^{j}}} \frac{1}{x_k} \right).$$

We now use our assumption that

$$\sum_{\substack{k:k > \overline{j} \\ a_k^{j-1} = a_j^{j}}} \frac{1}{x_k} - \sum_{\substack{k:k > j \\ a_k^{j-1} = a_j^{j-1}}} \frac{1}{x_k} < 0.$$

This gives us

$$\begin{aligned} \Sigma C_j^n(B) &= \Sigma C_j^n(A^{j-1}) + \delta \left(\frac{x_j}{x_k} + \sum_{\substack{k:k > j \\ a_k^{j-1} = a_j^{j-1}}} \frac{1}{x_k} - \sum_{\substack{k:k > j \\ a_k^{j-1} = a_j^{j}}} \frac{1}{x_k} \right) \\ &< \Sigma C_j^n(A^{j-1}) + \delta \frac{x_j}{x_k} < \Sigma C_j^n(A^{j-1}). \end{aligned}$$

But $\Sigma C_j^n(A^{j-1}) = \Sigma C_j^n(A^0)$, thus the profile B has a lower social cost as the social optimum. Hence equation (3) is correct and $\Sigma C_j^n(A^n) = \Sigma C_j^n(\mathcal{A}_2(A^0)) = \Sigma C_j^n(A^0)$. □

For arbitrary machines, the upper bounds are worse as in the case of the unnormalized objective functions. The following lemma provides an overview of the upper bounds:

Lemma 4. *Let $G = \langle N, M, (x_j), (s_i), C^P \rangle$ be a load balancing game. Then*

$$\mathrm{CR}(C_{\max}^n, G) \leq n \frac{\max_{i \in M} s_i}{\sum_{i \in M} s_i} + \frac{\max_{i \in M} s_i}{\min_{i \in M} s_i} \qquad \textit{and}$$

$$\mathrm{CR}(C_{\max}^n, G) \leq n \frac{\max_{i \in M} s_i}{\sum_{i \in M} s_i} + \frac{\max_{i \in M} s_i}{\min_{i \in M} s_i}.$$

Using the same reasoning as in the proof of Lemma 1 this lemma may be proved. We omit the details here.

5 Conclusions and Further Work

We have shown that with a slight increase in the coordination ratio for the objective function $C_{\max}$ our model performs much better for the other three objective functions. For the average completion time and the normalized average

completion time the decentralized stable solutions are as good as social optima in the case of identical machines. The drawback of the new model is that more information is required. The precedence relation $\preceq$ has to be announced as one of the rules of the game. Getting rid of the precedence relation or excluding the precedence relation from the game rules would remove this drawback compared to the KP-model.

One way to remove the precedence relation from the rules would be to choose a valid precedence relation randomly after starting the game and using the expected completion time instead of the real completion time for strategy selection. But this would make the situation worse, because the Nash equilibria would change. In concrete this would lead to a coordination ratio > 1 in the case of the (normalized) average completion time and identical machines. Further work is required to investigate if there are constant bounds for the coordination ratios in this randomized priority model.

A drawback of both the KP-model and our priority model is that only offline-situations are considered. Further work should include online-situations where players may take part in the game at different points of time and players do not know when a new player will begin to take part of the game. This would include other techniques as the classical games in normal form (or strategic games). By definition in strategic games the whole set of rules (number of players, allowed actions of all players and payoff functions of all players) are known to each player. This is not the case in the online situation where a player does not know when and if a further player will join the game. The advantage of the online variants would be that they model the realistic case more appropriately. A corresponding extension to network flow games (load balancing games are just a special case of them) would allow to model big decentralized networks like the Internet in a realistic way which is impossible with the offline models.

References

1. Berenbrink, P., Goldberg, L., Goldberg, P., Martin, R.: Utilitarian resource assignment. Computing Research Repository (CoRR) cs.GT/0410018 (2004)
2. Blum, N.: Personal communication. Rheinische Friedrich-Wilhelm-Universität Bonn, Informatik V (2005)
3. Borel, E.: La theéorie du jeu et les equations intégrales à noyau symétrique. Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences (Paris) 173 pp. 1304–1308 (1921)
4. Borel, E.: The theory of play and integral equations with skew symmetric kernels (translation of [3]). Econometrica 21 pp. 101–115 (1953)
5. Conway, R., Maxwell, W., Miller, L.: Theory of Scheduling. Addison Wesley (1967)
6. Czumaj, A., Vöcking, B.: Tight bounds for worst-case equilibria. Proc. of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) pp. 413–420 (2002)
7. Feldmann, R., Gairing, M., Lücking, T., Monien, B., Rode, M.: Nashification and the coordination ratio for a selfish routing game. In: Proc. of the 30th Internaltional Colloquium on Automata, Languages and Programming (ICALP). Lecture Notes in Computer Science (LNCS), vol. 2719, pp. 514–526 (2003)

8. Feldmann, R., Gairing, M., Lücking, T., Monien, B., Rode, M.: Selfish routing in non- cooperative networks: A survey. In: Proc. of the 28th International Symposium Mathematical Foundations of Computer Science. Lecture Notes in Computer Science (LNCS), vol. 2747, pp. 21–45. Spinger Verlag, Heidelberg (2003)
9. Fotakis, D., Kontogiannis, S., Koutsoupias, E., Macronicolas, M., Spirakis, P.: The structure and complexity of nash equilibria for a selfish routing game. In: Proc. of the 29th Internation Colloquium on Automata, Languages and Programming (ICALP'02). Lecture Notes in Computer Science (LNCS), vol. 2380, pp. 123–134 (2002)
10. Gairing, M., Lücking, T., Mavronicolas, M., Monien, B., Spirakis, P.: Extreme nash equilibria. In: Proc. of the 8th Italian Conference on Theoretical Computer Science (ICTCS). Lecture Notes in Computer Science (LNCS), vol. 2841, pp. 1–20 (2003)
11. Graham, R.: Bound for certain multiprocessor anomalies. Bell System Technical Journal 45, 1563–1581 (1966)
12. Immorlica, N., Li, L., Mirrokni, V.S., Schulz, A.S.: Coordination mechanisms for selfish scheduling. Theor. Comput. Sci. 410(17), 1589–1598 (2009)
13. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibira. In: Proc. of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS). Lecture Notes in Computer Science (LNCS), vol. 1563, pp. 404–413 (1999)
14. Liu, J.W.S., Liu, C.L.: Bounds on scheduling algorithms for heterogeneous comnputing systems. In: IFIP Congress 74, Stockholm. pp. 349–353 (1974)
15. von Neumann, J.: Zur theorie der gesellschaftsspiele. Mathematische Annalen 100, 295–320 (1928)
16. von Neumann, J.: On the theory of games of strategy (translation of [15]). Contributions to the Theory of Games, Volume IV (Annals of Mathematics Studies, 40) (1959)