CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
*Concurrency Computat.: Pract. Exper.* 0000; **00**:1–20
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe

# Coordination mechanisms for decentralized parallel systems

Johanne Cohen[1], Daniel Cordeiro[2], and Denis Trystram[34*]

[1] *Laboratoire de Recherche en Informatique (LRI, UMR 8623), Université Paris Sud; 91400 Orsay, France*
[2] *Department of Computer Science, University of São Paulo; Rua do Matão, 1010; 05508-090 São Paulo, Brazil*
[3] *LIG, Grenoble University; 51, avenue Jean Kuntzmann; 38330 Montbonnot, France*
[4] *Institut universitaire de France; 103, bd Saint-Michel; 75005 Paris, France*

SUMMARY

On resource sharing platforms, the execution of the jobs submitted by users is usually controlled by a centralized global scheduler. It determines efficient schedules regarding some common objective function that all organizations agree with (for instance, maximizing the utilization of the entire platform). However, in practice each organization is mostly interested in the performance obtained for its own jobs.
We study the price that the collectivity must pay in order to allow independence to selfish, self-governing organizations, so they can choose the best schedules for their own jobs. In other words, we are interested in analyzing the costs on the global performance inflicted by the decentralization of scheduling policies.
We present a game-theoretic model for the problem and the associated coordination mechanisms developed to reduce the cost of the decentralization of the decision-making process. The main contribution is to show (in theory and practice) how to devise pure Nash equilibria configurations for every instance of the problem and to prove that the price payed by the collectivity depends on the local scheduling policy and on the characteristics of the workload executed on such platforms. Copyright © 0000 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: scheduling; multiple organizations; algorithmic game theory; coordination mechanisms

## 1. INTRODUCTION

### *1.1. Multi-organizational computational platforms*

The scientific community has today the ability to combine different computational resources (possibly spread around the globe) into a powerful distributed system capable of analyzing massive data sets. These new computing platforms are currently empowering several e-Science projects, allowing different research organizations to team up and solve challenging large-scale scientific problems.

On such computing platforms, users belonging to different organizations are able to share their computational resources. An organization chooses to integrate a common computing platform not

*Correspondence to: Denis Trystram, LIG, Grenoble University; 51, avenue Jean Kuntzmann; 38330 Montbonnot, France. Tel.: +33 47 661 5264. E-mail: Denis.Trystram@imag.fr

Contract/grant sponsor: The São Paulo Research Foundation (FAPESP); contract/grant number: #2012/03778-0

only to contribute with its computational resources to other projects. Their users also expect to be able to execute their own jobs more efficiently with the help of the others.

This common setup works very well if all organizations have a common goal (that usually can be expressed as a single objective function for the entire community). However, in practice, these organizations may have slightly different performance objectives and are mostly interested in the performance obtained for their own jobs rather than the global performance that the community can achieve as a whole. A conflict of interests may arise if the resources of the cooperative platform are not shared in a fair manner. For example, if an organization presumes that its own jobs are being deferred by the central scheduler, then the organization may be compelled to leave the platform and to stop sharing its resources.

This selfish behaviour can hamper the scheduler's ability to compute the solution that optimizes the global objective and certainly can discourage the organizations to share their resources on these cooperative platforms. Cooperation between independent selfish organizations can be encouraged if each organization could have more control over the placement of its jobs and over the execution of the jobs assigned to its own machines.

This decentralization of the decision-making process of choosing where each job will be executed comes at a price. There is a clear trade-off between the global performance that can be obtained when we entrust a unbiased centralized entity to choose the best schedule that minimizes some common global objective, and the global performance that can be obtained when each organization is allowed to selfishly establish the best strategy for its own jobs, based only on some local performance metric. The decentralized approach usually produces solutions that are less efficient regarding the global objective. In this work, we study the price that the collectivity must pay in order to give independence to selfish organizations, so they can choose the best schedules for their own jobs.

### *1.2. Contributions and outline of the paper*

This article analyses the costs on the global performance inflicted by the decentralization of scheduling algorithms. We study the impact of the selfishness of each organization regarding the best global performance that could be obtained by truly unselfish organizations. The selfishness of each organization is expressed with two different characteristics: (i) each organization is mostly interested in the performance of its own jobs, and (ii) each organization has full control over its own machines, meaning that it may change the schedule devised by the centralized entity in favor of its own jobs.

We present a game-theoretic model for the problem of scheduling parallel rigid jobs on multi-core platforms for federated computing. Using tools from algorithmic game theory, we extend the notions of independence and selfishness of each participating organization. We first revisit previous results that were limited to sequential jobs [1, 2] and extend them to the more general case where the jobs are composed of parallel rigid jobs, showing that the characteristics of the workload actually defines how costly is the decentralization of the decision-making process.

The paper is organized as follows. Section 2 presents known results on the scheduling of parallel rigid jobs on multiple machines and how game theory has been applied to related problems.

The formal description of the scheduling problem studied on this work and the notations used are presented in Section 3 and the notion of selfishness in organizations in Section 4. This selfish behaviour motivated the study decentralization of the decision making, as described in Section 5

Section 6 shows that no pure $\epsilon$-approximate equilibrium (with $\epsilon \leq 2$) is possible using strategies based on classical list scheduling algorithms. However, in Section 7 we show that when the strategy also considers information about the organization that owns the jobs, then it is always possible to make the game converge to an equilibrium state. An interesting outcome is that the cost of decentralization actually depends on the workload.

In Section 8 we present an experimental analysis of our algorithm to compute an approximated pure equilibrium, showing that our solutions can optimize the overall usage of the cooperative platform to a great extent, while — most of the times — allowing an organization to achieve better individual solutions than it otherwise would.

Finally, Section 9 summarizes the obtained results and presents some discussions.

## 2. RELATED WORK

The efficient execution of parallel rigid jobs [3] (i.e., parallel jobs that may need more than one processor to be executed and that does not change their parallelism at runtime) is a well-known problem among the practitioners of High Performance Computing. The job scheduling is most usually a centralized decision process targeting the minimization of the maximum completion time (makespan) of the jobs. When parallel jobs must be scheduled on consecutive processors (like a rectangle), the problem relates to the classical Multiple Strip packing problem [4, 5]. Bougeret *et al.* [6] presented an asymptotic FPTAS, i.e., $(1+\epsilon)$-approximation with an additive constant and running-time polynomial in the size of the instance and in $1/\epsilon$. When the jobs can be scheduled on non-contiguous processors, the problem is equivalent to the problem of scheduling on Grid Computing platforms. Schwiegelshohn *et al.* [7] studied the problem both for the offline and online (non-clairvoyant) cases. They have developed two algorithms with approximation ratios of 3 and 5 regarding the makespan, respectively. Several works (see, for instance, [8, 9]) provide also non-guaranteed heuristics for the problem of resource management on federated platforms, but are not the focus of this work.

The individualism of organizations sharing a common infrastructure was first studied by Pascual *et al.* [10, 11]. They have extended the scheduling problem for the case where resources and jobs are shared by *independent organizations*, each of which having local performance objectives for their jobs besides the global makespan. Their main contribution is the analysis of a centralized 3-approximation algorithm for the makespan that always incite these organizations to cooperate.

The concept of selfishness on individualist organizations has been broaden by Cohen *et al.* [12]. Studying workloads of bag-of-tasks (sequential) jobs, they have analyzed situations where selfish organizations could change the schedule of the jobs assigned to its own machines. Their model introduces a novel *selfishness restriction*, where configurations that allow one organization to *cheat* the devised global schedule by re-inserting its jobs earlier in the local schedules are forbidden. The authors showed that when all organizations behave selfishly, any approximation algorithm has a ratio greater than or equal to $2 - \frac{2}{N}$ regarding the optimal makespan with local constraints and presented several 2-approximation algorithms for the global makespan that always respect the selfishness restriction.

The previous works studied the MOSP problem using classical combinatorial optimization approaches based on centralized scheduling algorithms. The first attempt to study a distributed approach (using Algorithmic Game Theory [13]) for the problem is due to Cohen *et al.* [2], who studied the problem for the restricted case of sequential, *bag-of-tasks* jobs; their results will be discussed in depth in Section 5.

Scheduling problems are typically modeled as *selfish load balancing* [14], with independent selfish and non-colluding agents. Load balancing games can always be expressed as potential games [15] and, therefore, always induce a pure Nash equilibrium. These results were further used and extended by other works on scheduling, which studied games where each job is in charge of a different player and the cost function is defined to be the completion time of the player's job. See, e.g., [14, 16, 17, 18, 19, 20]. Please refer to [21] for a comprehensive summary of these results.

One of the main interests of load balancing games is to formally study the inefficiency caused by the lack of coordination (i.e., the lack of a centralized control) of the agents. This inefficiency is captured by the concept of *price of anarchy*, that compares the worst-case performance of a Nash equilibrium to the cost of the optimal (social cost) solution. Christodoulou *et al.* [22] note, however, that unlike the competitive ratio (that expresses lack of information) and approximation ratios (that expresses the lack of resources) the price of anarchy is not enough to suggest a framework in which coordination algorithms for selfish agents should be designed and evaluated. They propose the notion of *coordination mechanisms*.

A coordination mechanism is a set of local policies, one for every machine, that specify how the jobs assigned to this machine will be scheduled, independently of how the other jobs were assigned to other machines. The goal is to reduce the price of anarchy by connecting the local cost with the social cost. Christodoulou *et al.* [22] studied the well-known LPT (Largest Processing Time) algorithm. Immorlica *et al.* [19] extended these results by studying other non-preemptive local policies. They analyzed the existence of pure Nash equilibria under these policies by proving that the game is a potential game. Dürr *et al.* [21] focus on a preemptive local policy called EQUI. They proved that such games have a pure Nash equilibrium with a different kind of machine environment. Moreover, when using EQUI, the game has a strong Nash equilibrium (in which no group of agents can cooperate and change its strategy in such a way that all agents in the group strictly decrease their costs).

## 3. THE MULTI-ORGANIZATION SCHEDULING PROBLEM

### *3.1. Definition*

MOSP (the Multi-Organization Scheduling Problem) [10] was first studied on the context of scheduling on Grid Computing platforms. Grid computing platforms are typically organized as a federated system where users and computational resources, belonging to different administrative domains (or organizations), share resources and exchange jobs with each other in order to simultaneously maximize the profits of the collectivity and their own interests. Organizations are free to join or leave the platform at any time if they feel that their expectations are not being fulfilled.

MOSP studies the dichotomy between the local and global objectives of the organizations participating on federated platforms. On the one hand, it is crucial to determine schedules that optimize the allocation of the jobs for the whole platform in order to achieve good system performances. On the other hand, it is important to guarantee the performance perceived by each organization in order to provide incentive for all organizations to collaborate.

We study a scheduling model for the problem in which different organizations own a physical cluster of identical machines that are interconnected. All organizations intent to minimize the total completion time of all jobs (i.e., the global *makespan*) while they individually intent to minimize the completion time of their own jobs.

Although each organization accepts to cooperate with others in order to minimize the global makespan, we assume that individually it behaves in a selfish way. We assume that whenever possible, each organization will prioritize its own jobs, scheduling and executing them before any job from other organizations.

More importantly, a solution for the MOSP problem must guarantee that no organization will be aggrieved by the final schedule. This additional constraint imposed by MOSP guarantees that all organizations will always have incentive to cooperate by assuring that the local makespan obtained by each organization is at least as good as the makespan that the organization could have obtained using only its own resources (its *local makespan*). This restriction imposed on the final schedule is called MOSP's *local constraint*.

Formally, we define the target platform as a parallel computing system with $N$ different organizations interconnected by a middleware. Each organization $O^{(k)}$ $(1 \leq k \leq N)$ has $m^{(k)}$ processors which can be used to run jobs submitted by users from any organization.

Each organization $O^{(k)}$ has $n^{(k)}$ parallel rigid jobs to be executed. Each job $J_i^{(k)}$ $(1 \leq i \leq n^{(k)})$ requires exactly $q_i^{(k)}$ processors (from a same organization) to run and will use these processors for exactly $p_i^{(k)}$ units of time. In order to simplify the notation, we will describe a job using the ordered pair $(p_i^{(k)}, q_i^{(k)})$. We assume that all organization have enough processors to execute any of the jobs $(q_i^{(k)} \leq m^{(k)}, \forall i, k)$ and that no preemption is allowed, i.e., after its activation, a job runs until its completion at time $C_i^{(k)}$.

We denote the makespan of a particular organization $k$ by $C_{\max}^{(k)} = \max\limits_{1 \leq i \leq n^{(k)}} (C_i^{(k)})$. The makespan that the organization could obtain by scheduling its jobs only on its own processors is denoted by $C_{\max}^{(k)\, local}$. The global makespan for the entire computing platform is defined as $C_{\max} = \max\limits_{1 \leq k \leq N} (C_{\max}^{(k)})$.

The MOSP optimization problem is stated as follows:

$$\text{minimize } C_{\max} \text{ such that, for all } k \ (1 \leq k \leq N),\ C_{\max}^{(k)} \leq C_{\max}^{(k)\, local}$$

The organizational model of the MOSP problem and the notations used are illustrated on Figure 1. The impact caused by the local restrictions on the global makespan can be see on Figure 2. The figure shows that all the algorithms that respect MOSP's local constraints have at least a makespan equal to $\frac{3}{2}$ of the value of the optimal solution without constraints.

### 3.2. *Previous works*

The MOSP problem has been studied mostly using classical tools from combinatorial optimization. Most solutions are based on centralized algorithms that produce (guaranteed) solutions respecting MOSP local and selfish constraints.

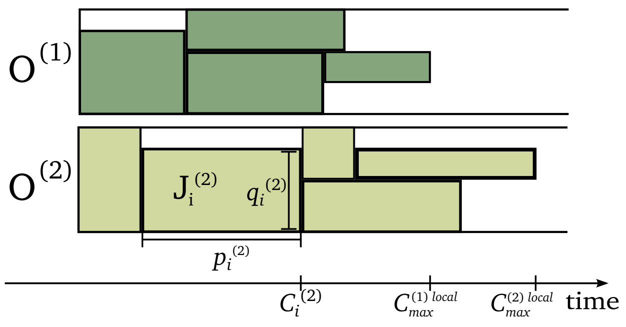
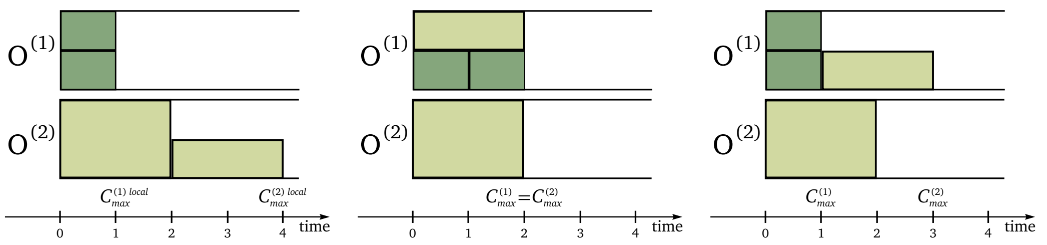


Figure 1. Example of an instance of the MOSP problem. The figure indicates two organizations ($O^{(1)}$ and $O^{(2)}$), their local makespan ($C_{\max}^{(1)\ local}$ and $C_{\max}^{(2)\ local}$) and a particular job $J_i^{(2)}$ that requires $q_i^{(2)}$ processors to be executed during $p_i^{(2)}$ units of time, finishing at time $C_i^{(2)}$. Two jobs of the same color indicates that they originally belong to the same organization.



(a) Initial instance.

(b) Optimal global $C_{\max}$ without MOSP's local constraints.

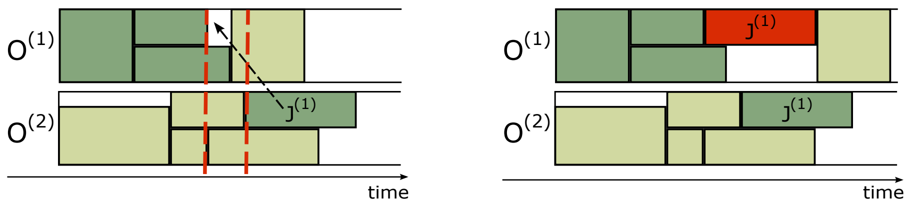
(c) Optimal global $C_{\max}$ respecting MOSP's local constraints.

Figure 2. Impact of MOSP's local constraints on the global makespan (adapted from [10]). In this instance (Figure 2a), the optimal global makespan that can be obtained for these jobs is equal to 2 (Figure 2b), but $O^{(1)}$ must have its makespan increased regarding the initial instance. The optimal schedule that guarantees that no organization will be penalized for cooperating (i.e., respects its the local constraint) is equal to 3 (Figure 2c).

MOSP was studied under different assumptions about the parallelism of the jobs. For parallel rigid jobs (i.e., that requires a fixed number of processors $q_i^{(k)}$), Pascual *et al.* [11] have showed a new algorithm with constant approximation ratio (a 3-approximation) regarding the optimal global makespan. They also show that this bound is asymptotically tight (when the number of organizations is large).

Cohen *et al.* [12] studied the same problem considering workloads restricted to *bag-of-tasks* applications, i.e., workloads consisting of sequential ($q_i^{(k)} = 1, \forall i, k$) jobs. In this case, they have showed how to adapt classical List Scheduling algorithms [23] to the MOSP problem while keeping the approximation ratio of $(2 - 1/N)$ provided by this kind of algorithm.

They have also explored two aspects of the selfishness of the organizations. First, they have showed that the best solutions that does not allow any organization to change the devised schedule are at most $(2 - 2/N)$ times worst than the optimal solutions that respect the classical MOSP constraints. Second, in a more detailed study [2], they proposed the first game-theoretic model for MOSP (still restricted to the case with sequential jobs). They first showed that games using Coordination Mechanisms based on classical scheduling algorithms like LPT or SPT, where the local schedules are given by some priority-rule based on some characteristic of the jobs, does not always admit a pure Nash equilibrium. If the priority-rule is also based on the organization that owns the jobs, then it is always possible to compute a pure $\rho$-approximate Nash Equilibrium.



(a) Example of schedule not enforcing the selfishness restriction.

(b) Selfish organization defrauding the example schedule.

Figure 3. Example of schedule forbidden by the selfishness restriction.

## 4. SELFISHNESS

The selfishness of the organizations that compose a cooperative platform can be expressed in different ways.

The first form of selfishness is directly related to the *local constraints* of the MOSP problem and to the nature of cooperative computing platforms. Any organization disappointed with the performance obtained after joining the platform always has the option to leave the platform and execute its own jobs by itself (using only its own computational resources). This act can have as side effect an increase on the global makespan of the jobs of the remaining organizations.

The second expression of selfishness considered in this work is related to the governance of the computational resources shared by the organizations. While the global schedule might be computed by a central entity, the organizations still are in control of how the jobs will be executed locally on its own machines. This particularity means that, in theory, it is possible for organizations to defraud the devised global schedule by re-inserting their jobs earlier in its own machines. This expression of selfishness was introduced by Cohen *et al.* [12].

In order to prevent such behavior, we have extended the classical MOSP problem with an additional restriction on the schedule called *selfishness restriction*. The idea is that in any schedule respecting this restriction, no organization can unilaterally improve its local makespan by changing how its own jobs will be executed on its own machines.

For the general case with workloads composed of parallel jobs, the selfishness restriction can be stated as follows. Given a fixed schedule, let $J_f^{(l)}$ be the first foreign job scheduled to be executed in $O^{(k)}$ (or the first idle time if $O^{(k)}$ has no foreign job) and $J_i^{(k)}$ any job belonging to $O^{(k)}$. Then, the *selfishness restriction* forbids any schedule where $C_f^{(l)} - p_f^{(l)} < C_i^{(k)} - p_i^{(k)}$ and $q_f^{(l)} \geq q_i^{(k)}$.

The selfishness restriction guarantees that no foreign job can start its execution on a machine belonging to organization $O^{(k)}$ if there still are remaining jobs from $O^{(k)}$ waiting to be executed on a later time on other organizations. Since we are considering parallel jobs, the only exception allowed to this rule is when the foreign job uses less processors than any job from $O^{(k)}$ waiting to be executed. In this case, these other jobs may not fit in the available local processors.

In other words, a selfish organization $O^{(k)}$ refuses to cooperate if one of its jobs could be executed earlier in one of $O^{(k)}$ machines even if this leads to a larger global makespan.

Figure 3 illustrates the selfishness restriction, showing how a schedule that does not enforce the selfishness restriction (Figure 3a) can be modified by a selfish organization (Figure 3b). In the example, organization $O^{(1)}$ is able to reduce its local makespan by executing a copy of job $J^{(1)}$ before the foreign jobs assigned to its own machines.

## 5. GAME-THEORETIC MODEL

The MOSP problem has been studied mostly using classical tools from combinatorial optimization. Most solutions are based on centralized algorithms that produce (guaranteed) solutions respecting MOSP local and selfish constraints. We study the interactions between the independent organizations as the result of rational selfish agents attempting to reach an equilibrium. This section describes MOSP modeled as a non-cooperative game.

We associate one selfish agent with each organization, i.e., jobs originally from organization $O^{(k)}$ are managed by agent $k$. Each agent can choose on which organization each one of its jobs will be executed, knowing that each selfish organization will schedule first its own jobs. Note that each agent is responsible for a set of jobs, while most of the previous works described in Section 2 assign one agent to each available job.

A pure strategy $S^{(k)}$ for agent $k$ is a vector of $n^{(k)}$ elements such that $S^{(k)}(i)$ corresponds to the organization chosen by player $k$ for job $J_i^{(k)}$. We denote $\mathcal{S}^{(k)}$ the set of all the strategies for agent $k$.

A configuration (or profile) $M$ is a vector $(S^{(1)}, S^{(2)}, \ldots, S^{(N)})$ such that $S^{(k)}$ is a strategy of agent $k$. The cost of an agent $k$ under configuration $M$ — denoted by $\text{cost}^{(k)}(M)$ — corresponds to the makespan obtained by organization $O^{(k)}$ ($C_{\max}^{(k)}$) on the schedule implied by the configuration $M$.

We are interested on studying situations where the choices of the agents lead to configurations where no agent has incentive to change its strategy. We say that this configuration is a *pure Nash equilibrium*. More formally, a configuration $M$ is a pure Nash equilibrium if all agent $k$, satisfies the following property: $\forall s \in \mathcal{S}^{(k)}$, $\text{cost}^{(k)}(M) \leq \text{cost}^{(k)}(s, M_{-k})$, where $M_{-k}$ is a vector $(S^{(1)}, S^{(2)}, S^{(k-1)}, S^{(k+1)} \ldots, S^{(N)})$.

The stability of an equilibrium is captured by the concept of *$\epsilon$-approximate equilibrium*. In $\epsilon$-approximate equilibrium, each selfish agent is satisfied when the chosen strategy results in an approximation of its best response. A configuration $M$ is a $\epsilon$-approximate equilibrium [24] if it holds that: $\forall s \in \mathcal{S}^{(k)}$, $\text{cost}^{(k)}(M) \leq \epsilon \times \text{cost}^{(k)}(s, M_{-k})$.

Every schedule has some social cost, as well as individual costs for every agent. The *social cost* of a configuration is the global makespan obtained on the system ($C_{\max}$). Due to the lack of coordination, configurations in equilibrium may have higher costs if compared to the global social optimum. A measure of this inefficiency is the *price of anarchy*. It is defined as the ratio between the cost of the worst Nash equilibrium and the optimal cost, which in general is not an equilibrium.

### *5.1. Local scheduling policy*

There are two novel aspects of the MOSP game if compared to the recent literature on game theory applied to scheduling problems (in particular, to the class of problems known as selfish load balancing problems). First, each agent is in charge of several jobs at once, instead of being responsible for only one job. Second, the agents choose the organization on which each job will be scheduled, but not on which processors on the parallel machine belonging to the organization. This implies that the set of strategies chosen by all agents to define where each job will be executed is not sufficient to compute the final schedule of all jobs. We need also a *coordination mechanism* [22] that defines how one organization will schedule and execute locally the jobs that were assigned to it.

This coordination mechanism must reflect the selfish behavior of each organization. In this work, we assume that all organizations are individualist and selfish. We also assume that each organization has full control on the scheduling of the jobs assigned to its machines, which means that, in theory, it is possible for organizations to cheat the devised global schedule by re-inserting their jobs earlier in the local schedules[†]. Therefore, a coordination mechanism for the MOSP game must consider that an organization will always prioritize the execution of its own jobs before any job owned by other organizations, no matter how this decision will impact the makespan of other organizations. In other words, each organization will apply a "my jobs first" policy, scheduling its own jobs before jobs migrated from other organizations.

The remaining (foreign) jobs must be scheduled according to some criteria. We added to the game model the notion of *job priority*. Organizations will compute the local scheduling of the foreign jobs according to their scheduling priority. These jobs will be scheduled in decreasing priority order, i.e., jobs with higher priority will be scheduled first and jobs with the same priority will be scheduled in no particular order.

The choice of this prioritization criteria determines the existence (or not) of pure Nash equilibria and their quality. A natural criteria is to use the information about the length $(p_i)$ and/or height $(q_i)$ of the jobs. This criteria is similar to classical list scheduling algorithms. Another possibility is to use also information about the organization that owns the jobs.

In the remaining sections, we will show how the choice of this criteria can determine the existence of pure Nash equilibria and its quality for the MOSP game. First, we study the MOSP game with *priority given to the jobs*, where all jobs have a distinct priority. Then, we study the game with *priority given to organizations*, where all jobs from a same organization have the same priority, but each organization has a distinct priority.

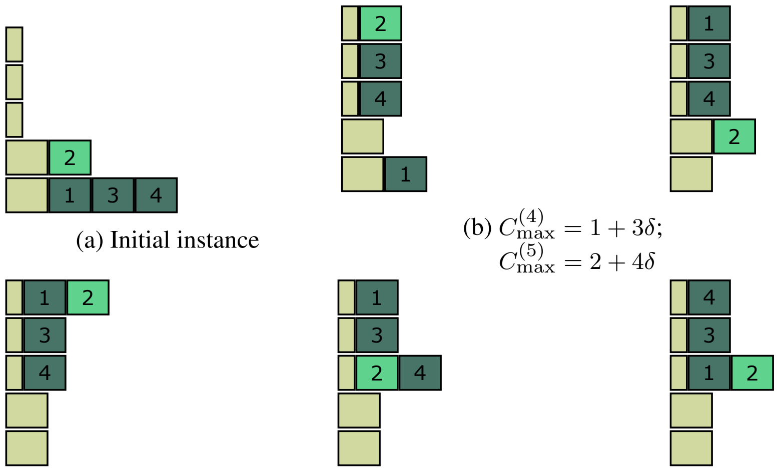
## 6. GAME WITH PRIORITY TO JOBS

Classical scheduling algorithms usually compute the order in which the jobs will be executed according to some rule that assigns a different priority calculated separately for each job. For instance, the Longest Processing Time first (LPT) scheduling algorithm prioritizes the jobs with larger processing times (the priority of each job is its length), the Shortest Processing Time first (SPT) algorithm prioritizes the jobs with smaller processing times, etc.

For the general scheduling problem, Immorlica *et al.* [19] analyzed games with coordination mechanisms based on some classical scheduling algorithms (with priorities given to jobs). They studied the scenario where each job is controlled by a selfish agent that selects the machine that minimizes the expected job completion time. Their study shows that the set of pure Nash equilibria can always be computed by the LPT algorithm if the coordination mechanism used is to prioritize the largest jobs (and, analogously, that the SPT algorithm calculates the set of pure Nash equilibria if the coordination mechanism prioritizes the smallest jobs).

Due to MOSP constraints, these results do not hold for MOSP games. With coordination mechanisms prioritizing jobs according to a criteria that does not depend on information about

[†]For a more detailed study about the impact on the quality of the obtained schedule caused by this selfish behaviour, please refer to [12].



(a) Initial instance

(b) $C^{(4)}_{\max} = 1 + 3\delta$; $C^{(5)}_{\max} = 2 + 4\delta$

(c) $C^{(4)}_{\max} = 2 + 4\delta$; $C^{(5)}_{\max} = 1 + 3\delta$

(d) $C^{(4)}_{\max} = 2 + 5\delta$; $C^{(5)}_{\max} = 1 + 3\delta$

(e) $C^{(4)}_{\max} = 1 + 3\delta$; $C^{(5)}_{\max} = 2 + 5\delta$

(f) $C^{(4)}_{\max} = 1 + 3\delta$; $C^{(5)}_{\max} = 2 + 5\delta$

Figure 4. Instance with no $\epsilon$-approximate equilibrium for $\epsilon < 2$.

the organization that owns the jobs, the MOSP game may not admit a pure Nash equilibrium, independently of the coordination mechanism being used.

We start by showing that with this priority model, even if we consider the notion of approximate equilibria, we cannot always have pure $\epsilon$-approximate equilibrium for values of $\epsilon < 2$.

### 6.1. *No Pure $\epsilon$-approximate Equilibrium for $\epsilon < 2$*

*Theorem 1*

MOSP games defined with a coordination mechanism that assign priorities to jobs independently of the owner organization do not admit a pure $\epsilon$-approximate equilibrium for values of $\epsilon < 2$.

*Proof*

We prove this theorem using the particular instance of the MOSP game composed only of sequential jobs ($q_i^{(k)} = 1$) depicted in Figure 4a. In this instance, organizations $O^{(1)}$, $O^{(2)}$, and $O^{(3)}$ have each one job of length equal to $\delta$, for some arbitrarily small constant $\delta > 0$. Organization $O^{(4)}$ has two jobs of length $1 + 2\delta$ and organization $O^{(5)}$ has four jobs also of length $1 + 2\delta$. We take an arbitrary coordination mechanism that prioritizes the jobs as follows: jobs $J_2^{(5)}$, $J_2^{(4)}$, $J_3^{(5)}$, and $J_4^{(5)}$ have priorities respectively equal to 1 (higher priority), 2, 3, and 4 (lower priority), while the remaining jobs have priorities lower than any of these four jobs. The priorities of these jobs are indicated in Figure 4.

MOSP local constraints impose that the low priority jobs $J_1^{(1)}$, $J_1^{(2)}$, and $J_1^{(3)}$ must be scheduled in their original organizations at time $t = 0$ (otherwise they would increase the makespan for their original organizations). Only organizations $O^{(4)}$ and $O^{(5)}$ are capable of improving their local makespans by changing the strategy for the higher priority jobs. The best makespan that organizations $O^{(4)}$ and $O^{(5)}$ can attain (while respecting MOSP's constraints) is equal to $1 + 3\delta$.

A best response policy for this game does not converge to an equilibrium. Figures 4d, 4e, and 4f show that both organizations can use their higher priority jobs to produce a configuration with the optimal makespan possible $(1 + 3\delta)$ by delaying a job from the other organization and increasing its makespan to $2 + 5\delta$.

Among all the possible configurations respecting MOSP's constraints, no configuration is a pure Nash equilibrium. Figures 4b and 4c show the only *Pareto-efficient* configurations for organizations $O^{(4)}$ and $O^{(5)}$, i.e., the configurations where the $C_{\max}$ of one organization cannot be improved without increasing the $C_{\max}$ of the other. When one of the organizations attain the optimal local makespan of $1 + 3\delta$, the best makespan that can be obtained by the other is $2 + 4\delta$.

In this example, the Pareto-efficient configurations give the smallest $\epsilon$ needed to obtain an approximated pure equilibrium. No pure $\epsilon$-approximate equilibrium exists unless $\epsilon \geq \frac{2+4\delta}{1+3\delta} \Rightarrow \epsilon \geq 2$.

This shows that MOSP games with coordination mechanisms that assigns priorities to jobs do not always admit a pure $\epsilon$-approximate equilibrium if $\epsilon < 2$. $\square$

Note that the result of Theorem 1 shows that there is a large class of different coordination mechanisms that do not admit $\epsilon$-approximate equilibrium for values of $\epsilon < 2$ for the MOSP game. Most notably, the theorem applies to games with coordination mechanisms based on classical scheduling algorithms like LPT, SPT, etc.

Theorem 1 shows that if the MOSP game is modeled with a coordination mechanism based on priority on jobs, then an $\epsilon$-approximate equilibria with $\epsilon < 2$ may not exist for some instances. An interesting question would be to known in advance if an instance admits at all a pure Nash equilibrium. Unfortunately, it is known [2] that for the MOSP game this problem is co-NP hard.

## 7. GAME WITH PRIORITY TO ORGANIZATIONS

Section 6 showed that when priorities are given individually to jobs, some instances may not have pure Nash equilibria, and even the existence of the "weaker" $\epsilon$-approximate equilibria is bounded to values of $\epsilon \geq 2$.

In this section, we study an alternative model for the assignment of priorities, where entire organizations are individually assigned with different scheduling priorities. In this model, each organization will locally schedule first its own jobs (the "my jobs first" policy) and then schedule each foreign job in non-increasing order of its owner priority. Two or more jobs belonging to a same organization are scheduled in no particular order.

Using this new game model, we present a parametric algorithm that computes a $\epsilon$-approximate equilibrium for any instance of the MOSP problem. Given a $\rho$-approximation list scheduling algorithm, the algorithm computes a schedule that is a $\rho$-approximate equilibrium.

We also study the impact on the quality of the global $C_{\max}$ obtained by the selfish organizations modeled by our game. We show that on the worst-case the price of anarchy is unbounded, but is asymptotically bounded by 2 for some specific workloads.

## *7.1. An algorithm to construct a pure $\rho$-approximate equilibrium*

In this section, we present a parametric algorithm that constructs a pure $\epsilon$-approximate equilibrium for instances of the MOSP with priority given to organizations. We show that using a $\rho$-approximation[‡] list scheduling algorithm for the classical $P||C_{\max}$ scheduling problem, we can always construct a pure $\rho$-approximate equilibrium configuration for the MOSP game.

The approximated pure equilibrium is constructed using a new load balancing scheme adapted from the algorithm known as Interactive Load Balancing Algorithm (ILBA) [26]. The idea of this adapted algorithm is to rebalance the load of the organizations from the organizations with higher priority to the organizations with lower priority.

We will denote the total order induced by the priorities on the organizations using the binary relation "$\succ$". If an organization $O^{(a)}$ has higher priority than an organization $O^{(b)}$, we will denote this fact by $O^{(a)} \succ O^{(b)}$.

The construction works as follows. First, each organization schedules all its own jobs locally using the given $\rho$-approximation scheduling algorithm. The organizations are then enumerated by non-increasing priority (i.e., for any two organizations $O^{(i)}$ and $O^{(j)}$, $i < j$ if and only if $O^{(i)} \succ O^{(j)}$).

All organizations $O^{(1)}, \dots, O^{(N)}$ are then rebalanced, one after the other, according to the priority order defined by the enumeration. The rebalancing of an organization $O^{(k)}$ is done by first unscheduling all jobs belonging to organization $O^{(k)}$, and then rescheduling all of them by executing the $\rho$-approximation list scheduling algorithm on all available processors on the platform. The algorithm must reassign a maximal set of jobs to the original organization, i.e., if at a given time the algorithm can choose between different organizations to schedule the job, then the original organization must be chosen.

At iteration $k$, the $\rho$-approximation algorithm will calculate a strategy for $O^{(k)}$'s jobs with cost no more than $\rho$ times the *optimal* solution. In particular, this means that: $\forall s \in \mathcal{S}^{(k)}$, $\text{cost}^{(k)}(M) \leq \rho \times \text{cost}^{(k)}(s, M_{-k})$, i.e., $O^{(k)}$ cannot improve its cost by a factor more than $\rho$.

We now show that:

*Theorem 2*
The algorithm presented in this section always produces a pure $\rho$-approximate equilibrium configuration for the MOSP game when priorities are given to organizations.

*Proof*
We prove the theorem by induction on $k$ (the index of the organization being rescheduled by the algorithm). We show that after rescheduling the jobs of organization $O^{(k)}$, no organization $O^{(i)}$ with $i \leq k$ can unilaterally improve its cost by a factor greater than $\rho$.

For $k = 1$, the algorithm will rebalance all jobs of organization $O^{(1)}$ applying the $\rho$-approximation algorithm. Even if $O^{(1)}$ has the highest priority among the other organizations, the "my jobs first" constraint does not allow any further unilateral improvement.

[‡]The factor $\rho$ of the algorithm used must have been analyzed for cases where machines starting times may be different from 0 like, for instance, Lee's MLPT [25] (Lee's MLPT provides a 4/3-approximation, while standard LPT provides a 3/2-approximation for sequential jobs). Any standard list scheduling algorithm have a guaranteed approximation ratio of 2.

Now assume that all organizations $1 \leq k \leq j-1$ cannot unilaterally improve their cost by a factor greater than $\rho$. We will show that after rescheduling the jobs from organization $O^{(j)}$, all already rescheduled organizations will not be able to improve their costs by a factor greater than $\rho$.

By contradiction, assume that an organization $O^{(i)}$ with $i < j$ can improve its makespan by a factor greater than $\rho$ by changing only its own strategy. In this case, $O^{(i)}$ must have had its makespan improved by migrating some of its jobs to organization $O^{(j)}$; otherwise, since $O^{(i)}$ has higher priority, it would have done that on some earlier interaction, contradicting the inductive hypothesis. So we can assume that at least one job from $O^{(i)}$ was migrated to $O^{(j)}$.

After rescheduling its jobs, if $O^{(j)}$ cannot improve its local makespan by migrating any job, then its jobs remain in its own organization. This means that any job from $O^{(i)}$ will start after the last job from $O^{(j)}$ because of the "my jobs first" constraint, which means that $O^{(i)}$ could have done that on an earlier iteration, which contradicts the inductive hypotheses.

On the other hand, if $O^{(j)}$ was able to improve its own makespan, then some of its jobs were migrated to other organizations. In particular, at least one of these jobs must have been rescheduled on a processor of a different organization to start either earlier or at the same time of the new value for the $C_{\max}^{(j)}$; otherwise this job would have been scheduled before at time $C_{\max}^{(j)}$. Let $P$ be this processor. If $O^{(i)}$ can improve its makespan by migrating one of its jobs to start at time at most $C_{\max}^{(j)}$ on a processor in organization $O^{(j)}$, then it could have improved even more if the job was migrated to processor $P$ in an earlier iteration, contradicting the inductive hypothesis that $O^{(i)}$ cannot improve its cost by a factor greater than $\rho$.

These two contradictions show that $O^{(i)}$ with $i < j$ cannot improve its makespan by a factor greater than $\rho$ by changing only its own strategy. This fact finishes the proof of the inductive step, showing that after rescheduling $O^{(j)}$, no already rescheduled organizations will be able to improve its cost by a factor more than $\rho$. □

Note that if the $\rho$-approximation algorithm given as parameter is the optimal, all organizations $O^{(k)}$ have no incentive to change their strategies. In other words:

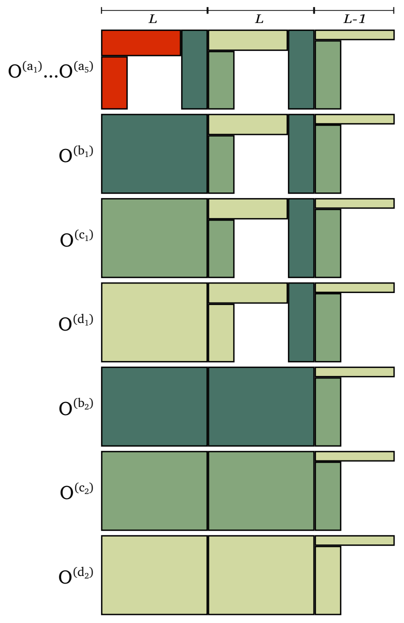
*Corollary 3*
The MOSP game with priority given to organizations always induces a pure Nash equilibrium.

The algorithm can be extended for the case where organizations have more than one processor available. A (potentially exponential) solution is to interactively reapply the algorithm on the obtained results until no agent can improve its makespan. We believe that an non-exponential algorithm for this case is still possible.

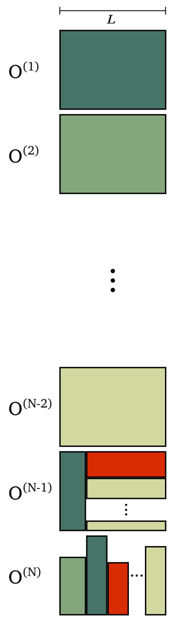
Previous results about the MOSP problem [12] indicates that computing the best response for each agent is NP-complete even if all jobs are sequential. Therefore, computing a pure Nash equilibrium remains a difficult problem.

## 7.2. *Price of anarchy*

Section 7.1 shows not only that all instances of the MOSP game admits a pure Nash equilibrium, but also how to compute it. The study of the existence of pure equilibria is interesting to show that there is at least a configuration where all selfish organizations are satisfied with the result. However, uncoordinated, selfish behavior can potentially lead to suboptimal social outcomes (in our case, to suboptimal results for the social cost, i.e., the global $C_{\max}$ of the system).



(a) Worst Nash equilibrium



(b) Optimal configuration

Figure 5. Example of the construction of a pure Nash equilibrium that leads to an unbounded PoA for $x = 2$. Each type of organization is depicted using a different color.

This effect is measured with the study of the *price of anarchy* of our game. The price of anarchy is defined as the ratio of the worst cost of an equilibrium to the optimal outcome.

We show that the price of anarchy of the MOSP game depends on the type of workload being considered. We start our analysis for the more general case, where the workload is composed of parallel jobs, and then refine the analysis to the particular case where the workload is composed of sequential, *bags-of-tasks* jobs.

*7.2.1. Parallel workloads* The MOSP problem models the scenario where each organization shares several machines — each of which having multiple processors (many-core machines) — and their users submit jobs that may need more than one processor to be executed.

For this more general scenario, we can show that:

*Theorem 4*
The price of anarchy in MOSP games with priority to organizations and parallel workloads is unbounded.

*Proof*
We will prove this theorem constructing an instance of the MOSP game with priorities to organizations that has a pure Nash equilibrium with cost arbitrarily far from the schedule with optimal cost. Like in Section 7, the construction of the pure Nash equilibrium is inspired by the ILBA [11] algorithm.

Let $x \in \mathbb{N}$ the ratio of the cost of this equilibrium to the ratio of the optimal cost and $n = \frac{3}{2}x(x-1)+2$. We will construct an instance of the MOSP game with $N = n + 3x$ organizations, all of which with $m$ identical processors.

The instance is composed of four types of organizations. We will denote these types as $\mathcal{A}, \mathcal{B}, \mathcal{C}$, and $\mathcal{D}$. The ith organization of each particular type will be denoted as $O^{(a_i)}$, $O^{(b_i)}$, $O^{(c_i)}$, and $O^{(d_i)}$, respectively.

We start defining the organizations of type $\mathcal{A}$. Let $L$ be an integer representing a large processing time (we will discuss the proper value of $L$ later.) Then we set $\mathcal{A}$ as a set of $n$ organizations, each one with two jobs[§]: one job $(L-1, x+1)$ and one job $(1, m-x-1)$.

For each $i \in \{1, \dots, x\}$, we define the organizations of types $\mathcal{B}$ and $\mathcal{C}$. An organization $O^{(b_i)}$ (of type $\mathcal{B}$) has $i$ jobs $(L, m)$ and $n + 3(i-1)$ thin jobs $(1, m)$, while an organization $O^{(c_i)}$ has $i$ jobs $(L, m)$, but has $n + 3(i-1) + 2$ slightly shorter jobs $(1, m - x + i - 1)$.

Similarly, we define the organizations of type $\mathcal{D}$ as follows. For each $i \in \{1, \dots, x\}$, $O^{(d_i)}$ has $i$ jobs $(L, m)$, one job $(1, m - x + i - 1)$, and $n + 3i$ jobs $(L-1, x-i+1)$.

To complete the description of the instance of the MOSP game, we must define the priority relation between the organizations. We define the priorities of these organizations based on the type of the organization. Organizations of type $\mathcal{A}$ have higher priority than any other organization. For organizations of the remaining types, we set, for each $i \in \{1, \dots, x-1\}$, the following priority relation: $O^{(b_i)} \succ O^{(c_i)} \succ O^{(d_i)} \succ O^{(b_{i+1})}$ and $O^{(b_x)} \succ O^{(c_x)} \succ O^{(d_x)}$.

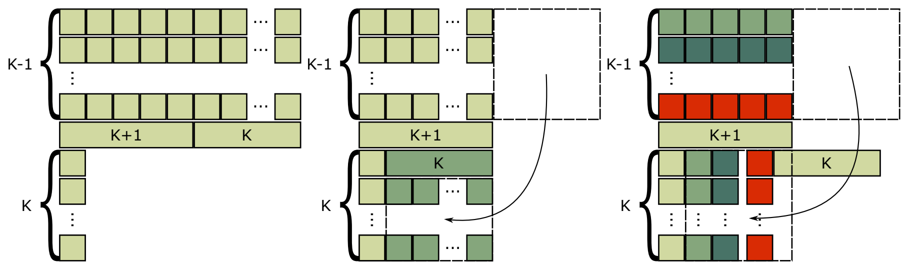
The worst pure Nash equilibrium is constructed using the following scheme. All $n$ organizations of type $\mathcal{A}$ do not have interest to move their jobs. They already have the smallest local makespan among all organizations. These organizations will only receive jobs from abroad. We focus on organizations $O^{(b_1)}$ and $O^{(c_1)}$. Organization $O^{(b_1)}$ has one job $(L, m)$ and $n$ jobs $(1, m)$. This organization keeps one job $(L, m)$ on its own machines and will migrate one other job to each organization of type $\mathcal{A}$ (that are less loaded than $O^{(b_1)}$). Moreover, organization $O^{(c_1)}$ has one job $(L, m)$ and $n+2$ jobs $(1, m-x)$. This organization keeps one job $(L, m)$ and one job $(1, m-x)$ on its own machines. It will migrate its other jobs to the $n+1$ less loaded organizations. Now, we will focus on organization $O^{(d_1)}$. $O^{(d_1)}$ keeps its jobs of sizes $(L, m)$, $(1, m-x)$, and $(L-1, x-i+1)$. It will migrate its other jobs to the $n+3$ less loaded organizations. The height of these jobs were chose in such a way that the shortest thin job is slightly too tall to fit in any gap, but short enough to be scheduled with a long thin job from an organization of type $\mathcal{D}$. Each of the $n + 3i$ long thin jobs of organizations of type $\mathcal{D}$ will be migrated to a different organization (among the ones with higher priorities) and will be scheduled at the end. These jobs cannot be scheduled earlier (inside a gap) because the processing time of these jobs are equal to $L-1$, while the gaps in the schedule have length $L-2$.

The remaining organizations $O^{(b_i)}$, $O^{(c_i)}$, $O^{(d_i)}$ and $O^{(e_i)}$, for each $i \in \{2, \dots, x\}$ are scheduled in a similar way, using the same construction.

All organizations have their local makespan improved (or kept unchanged) regarding the initial local makespan (i.e., MOSP's local constraints are respected) and no organization can unilaterally improve its local makespan because of the way the priorities were defined. Therefore the final schedule is a pure Nash equilibrium. Figure 5 illustrate one instance of the MOSP game (depicted with $x = 2$), with the configuration that is a pure Nash equilibrium (Figure 5a) and the optimal makespan (Figure 5b). The total cost (global makespan) of this schedule is $(x+1)L - 1$.

An optimal schedule for this instance, with $C_{\max} = L$, can be build as follows. Each large job of size $(L, m)$ is scheduled on one of the first $N-2$ organizations. This is possible since the number of

[§]Recall from Section 3 that the ordered pair $(p, q)$ represents a parallel job with length $p$ that requires $q$ processors.



(a) Initial instance (b) Optimal configuration (c) Worst Nash equilibrium

Figure 6. Price of anarchy of the MOSP game with priority given to organizations.

such large jobs is $3\sum_{i=1}^{x}$. This number is exactly equal to $N-2 = n+3x-2 = \frac{3}{2}x(x-1)+3x = \frac{3}{2}x(x+1)$.

To achieve the optimal makespan of $L$, the remaining jobs must be scheduled on the last two organizations. One of these organizations will process one job of size $(1, m)$ and all the jobs with processing time equal to $L-1$. The sum of the height of these jobs are equal to $H = \sum_{i=0}^{x}(n+3i)(x-i+1)$. Choosing $m = H$, we have that the makespan of this organization will also be equal to $L$.

Finally, we schedule all the remaining jobs on the last organization. Each of these jobs has a processing time equal to one. Setting $L$ to be equal to the number of these jobs, we have that the makespan of this organization will also be equal to $L$.

Consequently, the makespan resulting from this schedule is exactly equal to $L$. Since the schedule is compact, this schedule is optimal. This shows that the price of anarchy of this instance of the MOSP game is equal to $\frac{(x+1)L-1}{L} = x + \frac{L-1}{L}$. Since $x$ can be any positive integer, we can choose $x$ to be as large as wanted. In other words, the price of anarchy for this instance is unbounded. □

*7.2.2. Workloads of bags-of-tasks* Theorem 4 shows that the social outcome obtained by the selfish agents of the MOSP game can be as large as wanted if the workload is composed of parallel jobs.

In this section, we refine the analysis of the price of anarchy for the MOSP game with priority to organizations for workloads of sequential jobs (also called bags-of-tasks). In practice, this type of workload is one of the most common computational models for the target parallel platforms. They are popular because they scale efficiently and are easy to program.

For this type of workloads, we can show that:

*Theorem 5*
The price of anarchy (PoA) of MOSP games with priorities given to organizations for bag-of-tasks jobs is lower or equal to $2 - \frac{1}{N}$, and this bound is asymptotically tight.

*Proof*
The upper bound of the price of anarchy follows straightforwardly from the fact that a Nash equilibrium has no idle time. Consequently, the Nash equilibrium can be seen as a solution computed by an arbitrary list scheduling algorithm. An arbitrary list scheduling algorithm is a $(2 - \frac{1}{m})$-approximation, where $m$ corresponds to the number of machines.

We now show that this bound is asymptotically tight. Take the instance of the MOSP game (with priority given to organizations) depicted in Figure 6a. This game has $K-1$ organizations having

$2K+1$ jobs of size 1, one organization having two jobs of size $K+1$ and $K$, and $K$ organizations having one job of size 1. All jobs are sequential ($q_i^{(k)} = 1$). In this game, $O^{(i)} \succ O^{(j)}$ if $i < j$.

Figure 6b shows the optimal configuration (profile) possible. Jobs originally starting after time $K+1$ (highlighted in the figure) are migrated to the last $K$ organizations. On this configuration, every organization (except the last $K$ ones) achieves a local makespan equal to $K+1$.

The worst Nash equilibrium possible is the configuration depicted in Figure 6c. In this configuration, jobs starting after time $K+1$ from organizations $O^{(1)}, \ldots, O^{(K-1)}$ are rescheduled on the last $K$ organizations in such way that a job migrating from organization $O^{(i)}$ is rescheduled at time $i+1$ on these machines. Different job colors on Figure 6c indicate the ownership of the jobs from organizations $O^{(1)}, \ldots, O^{(K-1)}$. The worst local makespan is given by organization $O^{(k)}$ and is equal to $1 + (K-1) + K = 2K$.

This instance shows that the price of anarchy for the MOSP game is equal to $\frac{2K}{K+1}$, which for large values of $K$ is asymptotically equal to 2. □

At first glance, assigning different priorities to organizations seems unfair. Even if MOSP local constraints are always respected for all organizations, an organization may never achieve its optimal value for the local makespan because of the chosen priorities. However, for workloads composed of batches of bags-of-tasks jobs, fairness can be achieved in practice by rotating the priorities of each organization at each batch scheduling.

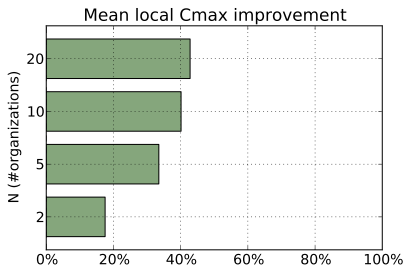
## 8. EXPERIMENTAL ANALYSIS

The price of anarchy of the MOSP problem with priority to organizations given by theorems 4 and 5 actually gives only an upper bound on how far from the optimal solution an equilibrium can be. Nonetheless, the price of anarchy does not give any evidence on the quality of the other possible equilibria.

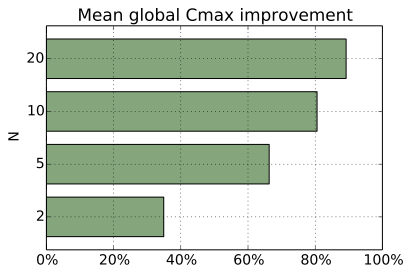
In this section, we present an experimental analysis on the quality of the approximated equilibria solutions obtained using the centralized parametric algorithm presented on Section 7.1. Recall from Theorem 2 that it is always possible to compute a Nash equilibrium and that our algorithm is able to compute an approximate equilibrium using any list scheduling algorithm.

Using the classic Longest Processing Time first (LPT [23]) algorithm, we conducted an extensive series of simulations with the proposed algorithm. The workload was randomly generated with parameters matching the typical environment found in academic grid computing systems [11]. We have tested instances with a diversity of numbers of organizations, machines, and jobs. For all combinations of $N \in \{2, 5, 10, 20\}$, $m \in \{32, 128, 512\}$ and total number of jobs $\sum_k n^{(k)} \in \{10, 50, 100, 500\}$, we have randomly generated jobs $(p_i^{(k)}, q_i^{(k)})$ with sizes and heights following a uniform distribution ranging from 1 to 50 and from 1 to $m$, respectively.

Each combination of these parameters was tested with 50 different random instances. In our tests, the number of initial jobs in each organization follows a Zipf distribution with exponent equal to 1.4267, which best models virtual organizations in real-world grid computing systems [27]. All results are presented with confidence level of 95%.



(a) Improvement on the local $C_{\max}$



(b) Improvement on the global $C_{\max}$

Figure 7. Mean percentage improvement on local and global $C_{\max}$ for $N = \{2, 5, 10, 20\}$.

Figure 7 shows the improvements obtained by our algorithm on the local $C_{\max}$ of the organizations and on the global $C_{\max}$ of the platform — *i.e.*, the mean values of ratios $\frac{C_{\max}^{(k)}}{C_{\max}^{(k)\ local}}$ and $\frac{C_{\max}}{\max_k C_{\max}^{(k)\ local}}$, respectively. Figure 7a shows that for platforms with larger number of organizations can achieve better improvements. For $N = 20$, the average improvement of individual organizations was 42.8%.

At first sight, the improvements on the global $C_{\max}$ are more expressive. The average improvement on the global $C_{\max}$ for $N = 2$ was 89.2%. Such huge improvement can be explained by the Zipf distribution of the jobs among the platform. It creates instances with heavy loaded organizations, that benefit greatly from the migrations of its jobs.

Notwithstanding the smaller gains, a more careful inspection on the results shows that the gains obtained are near (and even below) the theoretical lower bound for the $C_{\max}$ of each organization. Recall that for organization $O^{(k)}$, its local lower bound can be computed by $\max\{p_{\max}^{(k)}; \frac{\sum_{i=1}^{n(k)} p_i^{(k)} q_i^{(k)}}{m^{(k)}}\}$.

Figure 8 shows an histogram with the values of the initial makespan ($C_{\max}^{(k)\ local}$) and the local makespan obtained by our algorithm ($C_{\max}^{(k)}$) to the local theoretical lower bound of the organizations. It clearly shows that for all values of $N \in \{2, 5, 10, 20\}$, the makespan obtained by each organization on the approximated equilibrium is (most of the times) better than the original lower bound (corresponding to the value 1.0). This is possible because on the rebalancing phase of the algorithm, more machines are available for the organizations.

## 9. CONCLUDING REMARKS

On most of the works related to the scheduling of parallel rigid jobs on platforms for federated computing, the decision-making process is made by a centralized agent that computes a scheduling respecting some predefined rules. In this work, we have studied how to decentralize this processes, giving to the participants the freedom to choose the best strategy for their jobs. We have used algorithmic game-theory to extend the notions of independence and selfishness of each participating organization. The goal was to study the interactions between the independent organizations as the result of rational selfish players attempting to reach an equilibrium.

Our model represents the problem as a game, where each participant can choose on which organization each one of its jobs will be executed. The selfishness of each organization is expressed with two different characteristics: (i) coordination mechanisms that assume that each organization

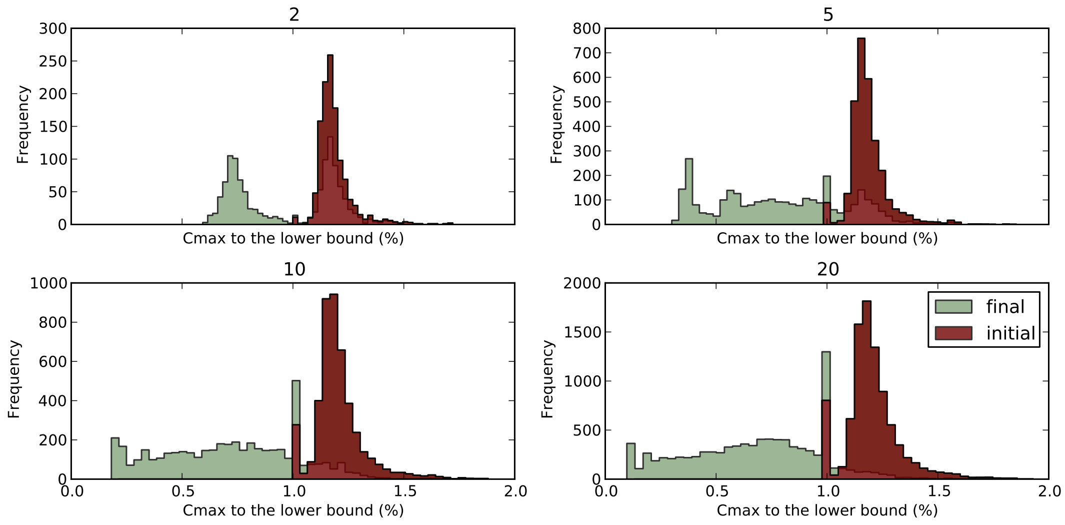


Figure 8. Histogram of the percentage improvement obtained by each organization regarding their initial and final $C_{\max}$ for $N = \{2, 5, 10, 20\}$.

will always execute its own jobs first and only after the jobs from other organizations (that in turn will be scheduled according some predefined local scheduling policy); and (ii) governance, considering the fact that each organization have full control of its own resources and have the power to change the schedule on its own machines if this can improve the performance for its own jobs.

If the local scheduling of the foreign jobs mimics classic list scheduling algorithms, fixing a priory for each job individually, then we found that no pure $\epsilon$-approximate equilibrium is possible for values of $\epsilon$ less than 2. In this case, it is known that even the problem of determining if an instance admits a pure Nash equilibrium is co-NP hard.

If it uses information about the organization that own the job to prioritize the jobs, then we can show how to make the game converge to an equilibrium state. We presented an algorithm that can compute an approximate pure equilibrium that produces results as close as wanted to a pure Nash equilibrium.

We found that the quality of the configurations that led to pure Nash equilibria depends on the characteristics of the jobs that compose the workload. For the general case of workloads composed of parallel rigid jobs, the price of anarchy — the ratio between the social cost (the global makespan) of a worst-case Nash equilibrium and the social cost of an optimal assignment — may be unbounded on the worst-case. Nevertheless, for workloads of sequential jobs, which are one of the most common computational models for the target parallel platforms, the price of anarchy is asymptotically equal to 2. In practice, we show that the results can be greatly improved.

This game-theoretic model opens the possibility for new future works to explore more coordination mechanisms for the problem. It would be interesting to study new coordination mechanisms leading to results with additional guarantees like fairness or a better social welfare, for instance. Another interesting research opportunity is to use this game-theoretic model to explore more altruistic ways of collaboration, where each organization could tolerate a controlled degree of degradation of its local performance in order to improve the cost for the collectivity.

## References

1. Cohen J, Cordeiro D, Trystram D, Wagner F. Multi-organization scheduling approximation algorithms. *Concurrency and Computation: Practice and Experience* 2011; **23**(17):2220–2234, doi:10.1002/cpe.1752.
2. Cohen J, Cordeiro D, Trystram D, Wagner F. Coordination mechanisms for selfish multi-organization scheduling. *Proceedings of the 18th International Conference on High Performance Computing*, HiPC, 2011; 1–9.
3. Feitelson DG, Rudolph L, Schwiegelshohn U. Parallel job scheduling – a status report. *Job Scheduling Strategies for Parallel Processing*, *Lecture Notes in Computer Science*, vol. 3277. Springer, 2005; 1–16.
4. Zhuk SN. Approximate algorithms to pack rectangles into several strips. *Discrete Mathematics and Applications* Jan 2006; **16**(1):73–85.
5. Ye D, Han X, Zhang G. On-line multiple-strip packing. *Proceedings of the 3rd International Conference on Combinatorial Optimization and Applications*, *Lecture Notes in Computer Science*, vol. 5573, 2009; 155–165.
6. Bougeret M, Dutot PF, Jansen K, Robenek C, Trystram D. Approximation algorithms for multiple strip packing and scheduling parallel jobs in platforms. *Discrete Mathematics, Algorithms and Applications* 2011; **03**(04):553–586.
7. Schwiegelshohn U, Tchernykh A, Yahyapour R. Online scheduling in grids. *IEEE International Symposium on Parallel and Distributed Processing*, 2008; 1–10.
8. Ranjan R, Harwood A, Buyya R. A case for cooperative and incentive-based federation of distributed clusters. *Future Generation Computer Systems* 2008; **24**(4):280–295.
9. Huang Y, Bessis N, Norrington P, Kuonen P, Hirsbrunner B. Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm. *Future Generation Computer Systems* 2013; **29**(1):402—415.
10. Pascual F, Rzadca K, Trystram D. Cooperation in multi-organization scheduling. *Euro-Par 2007 Parallel Processing*, *Lecture Notes in Computer Science*, vol. 4641/2007. Springer, 2007; 224–233.
11. Dutot PF, Pascual F, Rzadca K, Trystram D. Approximation algorithms for the multiorganization scheduling problem. *IEEE Transactions on Parallel and Distributed Systems* Nov 2011; **22**(11):1888–1895.
12. Cohen J, Cordeiro D, Trystram D, Wagner F. Analysis of multi-organization scheduling algorithms. *Euro-Par 2010 - Parallel Processing*, *Lecture Notes in Computer Science*, vol. 6272, D'Ambra P, Guarracino M, Talia D (eds.). Springer, 2010; 367–379.
13. Nisam N, Roughgarden T, Tardos E, Vazirani VV. *Algorithmic Game Theory*. Cambridge University Press, 2007.
14. Koutsoupias E, Papadimitriou C. Worst-case equilibria. *Computer Science Review* 2009; **3**(2):65–69.
15. Fotakis D, Kontogiannis S, Spirakis P. Selfish unsplittable flows. *Theoretical Computer Science* 2005; **348**(2):226–239.
16. Caragiannis I, Flammini M, Kaklamanis C, Kanellopoulos P, Moscardelli L. Tight bounds for selfish and greedy load balancing. *Algorithmica* 2011; **61**:606–637.
17. Czumaj A, Krysta P, Vöcking B. Selfish traffic allocation for server farms. *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, STOC, ACM: New York, NY, USA, 2002; 287–296.
18. Libman L, Orda A. Atomic resource sharing in noncooperative networks. *Telecom. Systems* 2001; **17**:385–409.
19. Immorlica N, Li LE, Mirrokni VS, Schulz AS. Coordination mechanisms for selfish scheduling. *Theoretical Computer Science* 2009; **410**(17):1589–1598.
20. Schuurman P, Vredeveld T. Performance guarantees of local search for multiprocessor scheduling. *INFORMS Journal on Computing* 2007; **19**(1):52–63.
21. Cohen J, Dürr C, Thang NK. Non-clairvoyant scheduling games. *Theory of Computing Systems* 2011; **49**(1):3–23.
22. Christodoulou G, Koutsoupias E, Nanavati A. Coordination mechanisms. *Automata, Languages and Programming*, *Lecture Notes in Computer Science*, vol. 3142, Diaz J, Karhumäki J, Lepistö A, Sannella D (eds.), Springer, 2004; 45–56.
23. Graham RL. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics* Mar 1969; **17**(2):416–429.
24. Radner R. Collusive behavior in noncooperative epsilon-equilibria of oligopolies with long but finite lives. *Journal of Economic Theory* 1980; **22**(2):136 – 154.
25. Guo-Hui L. The exact bound of Lee's MLPT. *Discrete Applied Mathematics* Jul 1998; **85**(3):251–254.
26. Pascual F, Rzadca K, Trystram D. Cooperation in multi-organization scheduling. *Concurrency and Computation: Practice & Experience* May 2009; **21**(7):905–921.
27. Iosup A, Dumitrescu C, Epema D, Li H, Wolters L. How are real grids used? The analysis of four grid traces and its implications. *7th IEEE/ACM International Conference on Grid Computing*, 2006; 262–269.