Tight Analysis of Relaxed Multi-Organization Scheduling Algorithms

Daniel Cordeiro, Pierre-Francois Dutot, Grégory Mounié, Denis Trystram

Tight Analysis of Relaxed Multi-Organization Scheduling Algorithms

Daniel Cordeiro*‡, Pierre-François Dutot‡, Grégory Mounié‡, and Denis Trystram†‡
*Instituto de Matemática e Estatística
Universidade de São Paulo
São Paulo, Brazil
†Institut universitaire de France
Paris, France
‡LIG, Grenoble University
Montbonnot Saint-Martin, France
{cordeiro,pfdutot,mounie,trystram}@imag.fr

***Abstract*—The goal of this paper is to study how limited cooperation can impact the quality of the schedule obtained by multiple independent organizations in a typical grid computing platform. This relaxed version of the problem known as the Multi-Organization Scheduling Problem (MOSP) models an environment where organizations providing both resources and jobs tolerate a bounded degradation on the makespan of their own jobs in order to minimize the makespan over the entire platform.**

More precisely, the technical contributions are the following. First, we improve the existing inapproximation bounds for this problem proving that what was previously though as not polynomially approximable (*unless* $P = NP$) is actually not approximable at all. We achieve this using two families of instances whose Pareto optimal solutions are on par with the previous inaproximability bounds.

Then, we present two algorithms that solve the problem with approximation ratios of (2; 3/2) and (3; 4/3) respectively. This means that when using the first (second) algorithm, if an organization tolerates that the completion time of its last job cannot exceed twice (three times) the time it would have obtained by itself, then the algorithm provides a solution that is a 3/2-approximation (4/3-approximation) for the optimal global makespan. Both algorithms are efficient since their performance ratio correspond to the Pareto optimal solutions of the previously defined instances.

***Keywords*-scheduling; multiple organizations; multi-objectives; approximation algorithms;**

I. Introduction

A. Motivation and Informal Presentation of the Problem

Grid computing systems allow unprecedented computational power by combining geographically dispersed computers into a single massively parallel distributed system. Users of such systems contribute with computational power (multi-core machines, clusters, etc.) and expect to be able to execute their own jobs more efficiently by sharing their own resources with others.

This work has been supported by the bilateral brazilian-french CAPES-COFECUB program (project # Ma 660/10) and by a Google Research Award.

The heterogeneity of the available resources, the large number of available processors and cores, and different demands from users make the problem of scheduling of such parallel platforms really hard in practice. In order to fully exploit such systems, we need sophisticated scheduling algorithms that encourage the users to share their resources and, at the same time, that respect each user's own interests.

Typically, these computer systems are composed of organizations that own and manage clusters of computers. A user of such systems submits his/her jobs to a scheduler that can choose any available machine in any of these clusters. However, each organization expects that the performance of its own jobs will be improved even if its local hardware could be used to execute jobs from other organizations.

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 an incentive to the cooperation between organizations. The goal of this paper is to study how to compute such schedules and to analyze the trade-off between the global performance and the performance perceived by each organization.

B. Related Work

The classical problem of scheduling parallel jobs is related to the Strip packing [1] problem, where one must pack a set of rectangles (without rotations and overlaps) into a strip of processors in order to minimize the used height (which corresponds to the makespan). Strip packing was extended to the case where the rectangles must be packed into a finite number of strips [2], [3]. Recently, Jansen et al. [4] presented an asymptotic $(1+\epsilon)$-approximation AFPTAS with additive constant $O(1)$ and with running-time polynomial in n and in $1/\epsilon$.

Motivated by grid computing systems, Schwiegelshohn, Tchernykh, and Yahyapour [5] studied a very similar problem, with the difference that the jobs can be sched-

uled in non-contiguous processors instead of rectangles. They presented an algorithm with a guarantee that is a 3-approximation for the maximum completion time (makespan) when all the jobs are known in advance. They also provided a 5-approximation for the makespan on the on-line, non-clairvoyant case.

The Multi-Organization Scheduling problem (MOSP) was introduced by Pascual et al. [6], [7]. The goal was to study how to efficiently schedule parallel jobs in new computing platforms, while respecting users' own selfish objectives. A preliminary analysis of the scheduling problem on homogeneous clusters (composed of the same amount of identical processors) was presented with the target of minimizing the makespan, resulting in a centralized 3-approximation algorithm.

Then, MOSP was extended in distinct ways. Cohen et al. [8] analyzed the complexity of the problem when organizations are locally interested in minimizing either the makespan (MOSP($C_{\max}$)) or the average completion times (MOSP($\sum C_i$)). The problem is shown to be NP-Hard for both local objectives. This study was restricted to sequential jobs. In the same paper, they have also studied the impact of the lack of cooperation between organizations through the introduction of the notion of *selfish organizations*, i.e., organizations that refuse to cooperate if their objectives could be improved just by executing early one of their jobs in one of their own machines. Cohen et al. proved that any approximation algorithm for MOSP($C_{\max}$) has a ratio greater than or equal to $2 - \frac{2}{N}$ regarding the optimal makespan *with local constraints* if all organizations behave selfishly. Three different 2-approximation algorithms for MOSP with selfishness restrictions were presented and analyzed for practical workloads.

Ooshita et al. [9] also studied the notion of cooperation between organizations, following [6] but with a different perspective. Instead of studying the strict lack of willingness to cooperate, they studied how much one can improve the global makespan if all organizations allow some degradation of their own local objectives by a constant factor – called *degree of cooperativeness* (denoted by the parameter α). This relaxed version of MOSP was called α-*Cooperative Multi-Organization Scheduling Problem* (α-MOSP). The authors have focused their work on the problem of scheduling on unrelated machines ($R||C_{\max}$), opposed to the previous works on MOSP that always studied the problem of scheduling on independent machines ($P||C_{\max}$).

Their first contribution was to show that for any degree of cooperativeness $\alpha \geq 1$, there exists an instance of α-MOSP where the ratio between the makespan that respects the degree of cooperativeness ($C^{\alpha}_{\max}$) and the optimal makespan without the local constraints ($C^{*}_{\max}$) satisfies the relation $\frac{C^{\alpha}_{\max}}{C^{*}_{\max}} \leq \max\left\{\sum_{l=1}^{N} \frac{1}{\alpha^l}; \frac{(\alpha+1)}{\alpha}\right\} - \epsilon$. This ratio shows that when $\alpha = 1$ the lack of cooperation can make the makespan be $N-\epsilon$ times greater then it could have been with unlimited cooperation.

The authors developed an algorithm called ToMoS, which provides a way to transform a schedule with unrestrained cooperation into one with degree of cooperativeness α. The algorithm guarantees that the ratio between the $C_{\max}$ of the solution without the constraints and the solution constructed with the α-MOSP constraints is less then or equal to $\sum_{l=1}^{N} \frac{1}{\alpha^l} < \frac{\alpha}{(\alpha-1)}$.

The authors have studied the complexity and inapproximation bounds for the α-MOSP problem. They have showed that the problem is strongly NP-Hard for any $\alpha > 1$. Using the arguments given in the reduction used on the proof, they have calculated the inapproximation bounds for the problem when $\alpha < 2$. Under the assumption of $\mathbf{P} \neq \mathbf{NP}$, there is no ρ-approximation algorithm for α-MOSP for any $\rho < \frac{(\alpha+1)}{\alpha}$. If $\alpha > 2$, the classical inapproximation ratio of $\frac{3}{2}$ of the ∞-MOSP problem holds.

The notion of cooperation between different organizations and the study of the impact of users' selfish objectives are directly related to Game Theory. The study of the Price of Anarchy [10] on non-cooperative games allows to analyze how far the social costs – results obtained by selfish decisions – are from the social optimum on different problems. In selfish load-balancing games (see [11] for more details), selfish agents aim to allocate their jobs on the machine with the smallest load. In these games, the social cost is usually defined as the completion time of the last job to finish (makespan). Several works studied this problem focusing in various aspects, such as convergence time to a Nash equilibrium [12], characterization of the worst-case equilibria [13], etc. We are not targeting here at such game theoretical approaches.

C. Contributions and Outline of the Paper

As emphasized in the last section, the classical MOSP has been studied in depth. In this work, we extend the previous analysis on the relaxed version of the problem when local degradations are allowed. We believe to have provided a better understanding of this problem. More precisely,

- We improved the existing inapproximation bounds of [9] for α-MOSP by showing that, unlike previously thought, there is no polynomial time approximation algorithm for these bounds *even if* $P = NP$. Then, we present two families of instances whose Pareto optimal points corroborate the presented inapproximation bounds;
- Then, we propose two new algorithms with guaranteed performance to solve the α-MOSP problem. The analysis shows that the first one achieves a $\frac{3}{2}$-approximation for the obtained global makespan, while it guarantees that no organization will have its makespan more than doubled. This solution is Pareto optimal according to [9]. The second one guarantees a $\frac{4}{3}$-approximation

for the global makespan, while no organization has its makespan more than tripled. This solution belongs to the border of the inapproximability of the second family, and, thus, it is Pareto efficient.

II. Problem Description and Notations

The general problem studied in this paper is the scheduling problem in which different *organizations* own identical machines that are interconnected. Like in any grid computing system, these organizations share resources and exchange jobs with each other in order to simultaneously maximize the profits of the collectivity and their own interests. 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 in a selfish way.

Although each organization accepts to cooperate with others in order to minimize the global makespan, individually it behaves in a selfish way. An organization could refuse to cooperate if in the final schedule its local makespan is significantly increased.

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

Each organization $O^{(k)}$ has $n^{(k)}$ jobs to execute. Each job $J_i^{(k)}$ $(1 \leq i \leq n^{(k)})$ will use one processor for exactly $p_i^{(k)}$ units of time. 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 $C_{\max}^{(k)} = \max_{1 \leq i \leq n^{(k)}} (C_i^{(k)})$. The global makespan for the entire grid computing system is defined as $C_{\max} = \max_{1 \leq k \leq N} (C_{\max}^{(k)})$.

A. Relaxed Local Constraints

MOSP – as first studied by Pascual et al. [6], [7] – introduces *local constraints* to guarantee that all organizations will always have incentive to cooperate. In the global schedule, no organization will have its makespan increased when compared to the makespan that the organization could have by scheduling its jobs alone in its own set of processors ($C_{\max}^{(k)\ local}$). More formally, we denote by **MOSP($C_{\max}$)** the following optimization problem:

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

By restraining the feasible schedules to the ones that respect the *local constraints*, the minimum attainable global $C_{\max}$ is restricted. There is a clear trade-off between how much each organization can improve its own local makespan and how much the global makespan can be improved. This motivated Ooshita et al. [9] to study a relaxed version of the MOSP problem called the α-*Cooperative Multi-Organization Scheduling Problem* (abbreviated by α-MOSP).

In the α-MOSP problem, the *local constraints* imposed in the classical MOSP problem are relaxed. Each organization allows a degradation of its initial makespan by a factor $\alpha \geq 1$ that represents the degree of cooperativeness. Ooshita et al. study how much the global makespan can be improved if the makespan obtained by each organization k is bounded by $\alpha C_{\max}^{(k)\ local}$. When $\alpha > 1$, each organization is less selfish and is more likely to sacrifice its local objective in order to improve the global makespan. When $\alpha = 1$, the problem corresponds to the classical MOSP problem defined in [6].

The MOSP optimization problem rewritten to model the degree α of cooperativeness can be stated as follows:

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

In this paper, we are interested in the study of algorithms with guaranteed approximation ratios when the degree of cooperativeness is fixed. We denote the approximation ratios by $(\alpha; \beta)$, meaning that if an algorithm respects a degree of cooperativeness of α, then the algorithm is a β-approximation for the global $C_{\max}$. We also present some improved inapproximability analysis for the problem.

III. Inapproximability Analysis

A natural question that arises when studying a multi-objective optimization problem – like the relaxed version of MOSP – is how to determine the Pareto set that characterizes the set of Pareto optimal solutions[1]. In this section we studied how to characterize the Pareto set of MOSP. While this problem is hard (and still open), we provide some inapproximability results that should help to better characterize the Pareto set.

We provide in this section some families of instances that clearly show the trade-off between the objectives being optimized. We will show through these examples that if we bound the approximation ratio of one criteria, no scheduling algorithm will be able to improve the approximation ratio of the other objective. Those inapproximability results are stronger than in [9] because the shape of the inapproximation curve is broader (see Family 1 of instances on Section III-B) and because we prove that there is no algorithm with better performance ratio (since the Pareto optimal solutions of these instances reach these ratios). Hence, while previous works show that no polynomial algorithms with better ratios exists unless $P = NP$, we show that there are no feasible solutions with better ratios at all, which eliminates the possibility of any further improvements even with the use of non-polynomial algorithms.

[1]Pareto optimality [14] is a concept originally used in economics and now widely utilized to indicate that a solution for a multi-objective problem cannot be improved on one objective without worsening another objective.

A. Principle

To better understand the inapproximation ratios presented in this section, we first start with a simple example. Let us consider an instance with $N = 3$ organizations and four different jobs, as follows (see Figure 1):

$O^{(1)}$: 1 organization with 2 jobs of length 1,
$O^{(2)}$: 1 organization with 1 job of length $\frac{1}{3}$,
$O^{(3)}$: 1 organization with 1 job of length $\frac{2}{3}$.

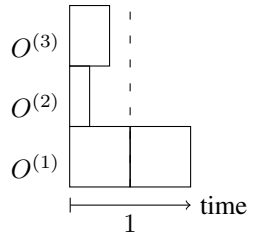
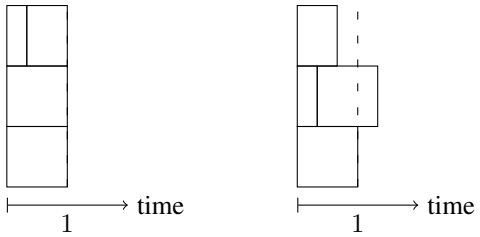


Figure 1. Simple instance with 3 organizations

First, remark that if we want to obtain a makespan strictly better than 2, then, it is impossible to schedule two jobs of organization $O^{(1)}$ on the same machine. Depending on the considered objective (global makespan and respect of the degree of cooperativeness α), a scheduling algorithm could be interested in either achieving the optimal global makespan ($C_{\max} = 1$) or respect α-MOSP relaxed local constraints with $\alpha = 1$ ($C_{\max}^{(k)} \leq 1 \cdot C_{\max}^{(k)\ local}, \forall k \in [1; N]$).

To achieve a makespan of 1, the jobs of organizations $O^{(2)}$ and $O^{(3)}$ must be scheduled together. In the best case, the job of $O^{(2)}$ is scheduled before the job of $O^{(3)}$ which leads to an approximation ratio of $(\frac{3}{2}; 1)$. This means that if the approximation ratio for the global makespan is bounded by 1, then no algorithm can construct a solution with a degree of cooperativeness better than $\frac{3}{2}$ (see Figure 2(a)).

On the other hand, if the schedule targets to achieve $C_{\max} < 2$ and a degree of cooperativeness $\alpha = 1$, then the jobs of $O^{(2)}$ and $O^{(3)}$ must start at time 0. Starting the first job of $O^{(1)}$ at time 0, the second one can start as soon as the job of organization $O^{(2)}$ finishes. In this case, the approximation ratio obtained is $(1; \frac{4}{3})$ (see Figure 2(b)).



(a) $(\frac{3}{2}; 1)$ (b) $(1; \frac{4}{3})$

Figure 2. The two relevant schedules

As these two schedules are Pareto optimal solutions for this instance, there is no algorithm with a performance ratio strictly better than $\frac{3}{2}$ on the degree of cooperativeness and $\frac{4}{3}$ on the makespan at the same time, as there are no solutions for this instance with these values.

The principles demonstrated in this example can be extended with the following instances.

B. Family 1

Let us consider the following instance of the MOSP problem (depicted in Figure 3):

- N organizations;
- $O^{(1)}$ has $n^{(1)} = N - 1$ identical jobs of length 1;
- For $2 \leq k \leq N$, $O^{(k)}$ has only one job of length $a_i = \frac{1}{N-1}$

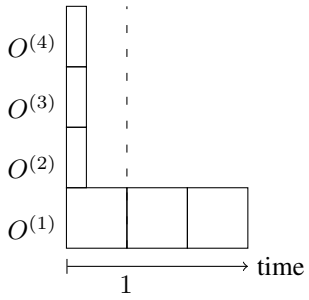
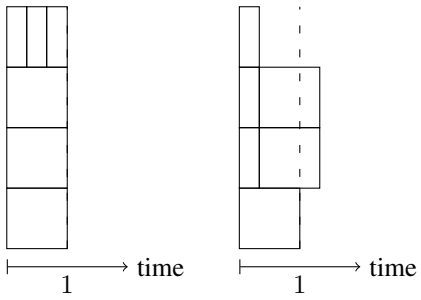


Figure 3. Instance of family 1 for $N = 4$

The total work to be done is equal to $\sum_{i,k} p_i^{(k)} = N$ and the optimal $C_{\max}$ for this instance *without the* MOSP *constraints* is obtained by scheduling each job of length 1 on different machines and then, scheduling all jobs of length $\frac{1}{N-1}$ on the remaining machine. The optimal $C_{\max}$ is equal to 1 (see Figure 4(a)).



(a) $(N - 1; 1)$ (b) $(1; 1 + \frac{1}{N-1})$

Figure 4. The two extreme Pareto schedules of family 1 for $N = 4$

Suppose now that we want to guarantee a value $1 \leq x < 2$ for the approximation ratio of the global makespan. This means that we must schedule all the jobs of $O^{(1)}$ in different organizations and that they must start at most at time $t = x - 1$.

If we set $x = 1 + \frac{1}{N-1}$, then we can schedule all the jobs of length a_i before the jobs of length 1 (see Figure 4(b)). This

leads to a $(1;\ 1+\frac{1}{N-1})$-approximation. On the other hand, if we set $x = 1+\frac{1}{N-1}-\epsilon$ then all jobs a_i must be scheduled on the machine that does not have any job of length 1. This schedule increases the makespan of the jobs a_i by a factor of at most $N-1$ and we get a $(N-1;\ 1+\frac{1}{N-1}-\epsilon)$-approximation.

This family of instances shows that although the guarantee $(N-1;\ 1+\frac{1}{N-1})$ seems far from the two Pareto optimal solutions $(1;\ 1+\frac{1}{N-1})$ and $(N-1;\ 1)$, we can not improve simultaneously the solution for both objectives.

For $N = 3, 4, 5$ and 6, we have Pareto efficient guarantees of $(2;\ \frac{3}{2})$, $(3;\ \frac{4}{3})$, $(4;\ \frac{5}{4})$, and $(5;\ \frac{6}{5})$, respectively.

C. Family 2

Consider the family of instances of the MOSP problem described as follows. Let j, k be integers such that $j > 1$ and $k > j-2$. We define three classes of organizations:

$O^{(A)}$: $(j-1)k$ organizations with only one job of length $\frac{j-1}{j}$
$O^{(B)}$: k organizations with $j-1$ jobs of length $\frac{1}{j}$
$O^{(C)}$: 1 organization with $k+1$ jobs of length 1

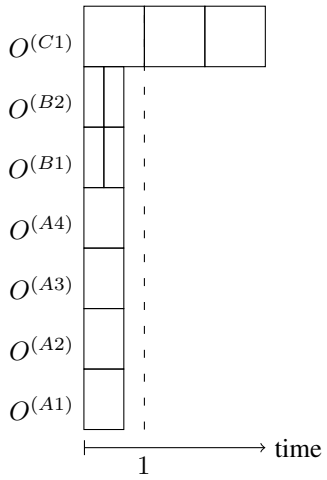


Figure 5. Instance of family 2 for $(j = 3; k = 2)$

In order to reach the optimal makespan, each job of organization $O^{(C)}$ must be scheduled alone on an organization. Each one of the jobs of $O^{(B)}$ must be scheduled together with a job from $O^{(A)}$, in such a way that each pair is scheduled alone on an organization and the job from $O^{(B)}$ is scheduled before the one from $O^{(A)}$. The global makespan is optimal ($C_{\max} = 1$) and the degree of cooperativeness is equal to $\frac{\frac{j-1}{j}+\frac{1}{j}}{\frac{j-1}{j}} = \frac{j}{j-1}$ (this configuration is shown in Figure 6(a)).

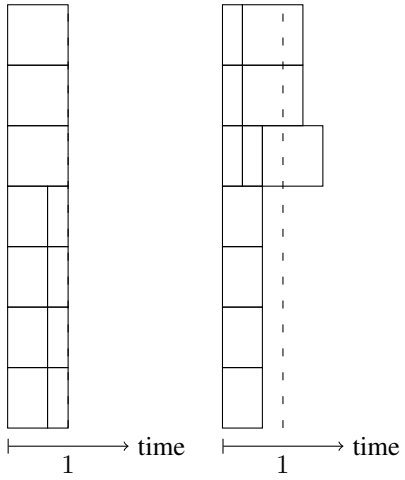
Proposition 1: To improve the degree of cooperativeness to a value better than $\frac{j}{j-1}$, the makespan must be at least equal to $1+\frac{j-1}{j}$.

Proof: To prove this, we need to look at what would be an implication of a lower degree of cooperativeness. If the degree of cooperativeness is lower than $\frac{j}{j-1}$, then each job of a type $O^{(A)}$ organization has to be scheduled without any other job of organizations $O^{(A)}$ or $O^{(B)}$. This only leaves $k+1$ machines to schedule the $k(j-1)$ jobs of organizations $O^{(B)}$. Furthermore, if the makespan is strictly lower than $1+\frac{j-1}{j}$, only one job of $O^{(C)}$ and at most $j-2$ jobs of $O^{(B)}$ can be scheduled on those $k+1$ machines (see Figure 6(b)).

However, as $k > j-2$, we have:

$$(k+1)(j-2) = k(j-2)+j-2 < k(j-2)+k = k(j-1)$$

Therefore, if the makespan is strictly lower than $1+\frac{j-1}{j}$ there is at least a machine with a job of type $O^{(B)}$ and one of type $O^{(A)}$, which lead to a cooperativeness ratio of at least $\frac{j}{j-1}$. ■



(a) $(\frac{j}{j-1}; 1)$ (b) $(1; 1+\frac{j-1}{j})$

Figure 6. Two Pareto schedules of family 2 for $(j = 3; k = 2)$

The Pareto guarantees are then $(\frac{j}{j-1};\ 1+\frac{j-1}{j})$. For $j = 2, 3, 4$ and 5, we have Pareto efficient guarantees of $(2;\ \frac{3}{2})$, $(\frac{3}{2};\ \frac{5}{3})$, $(\frac{4}{3};\ \frac{7}{4})$, and $(\frac{5}{4};\ \frac{9}{5})$, respectively.

D. Summary

In this section, we have studied the trade-offs between the degree of cooperativeness of the organizations and the best global makespan attainable that respects the relaxed MOSP constraints. We presented two families of instances and their respective sets of Pareto optimal approximation ratios.

The first family of instances, presented in Section III-B, improves the previous known bounds established by Ooshita et al. [9] in two distinct ways.

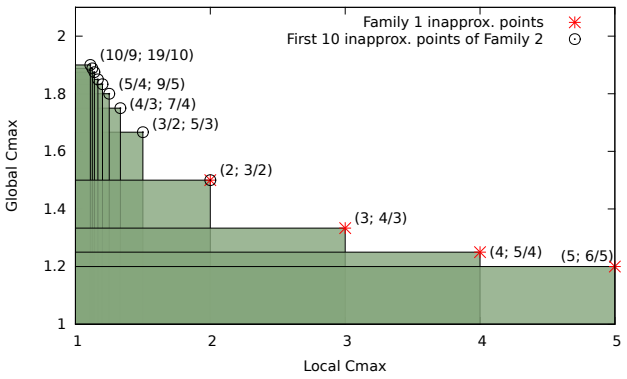


Figure 7. Inapproximation bounds

First, we proved that what was previously though as not polynomially approximable (*unless* $P = NP$) is actually not approximable at all. We show that for this family of instances it is not possible to simultaneously improve the guarantee of $(N;\ 1 + \frac{1}{N-1})$ on both criteria.

Second, despite the fact that the approximation ratios obtained in families 2 and 3 are in the curve $\frac{(\alpha+1)}{\alpha}$ presented in [9], better ratios are theoretically still possible if only one criterion is improved. The rectangles in Figure 7 mark the area that is known to be not attainable. However, points on the outline of the rectangles (that are not covered by other rectangles) are still attainable.

The second family of instances, presented in Section III-C, shows that ratios of the form $(\frac{j}{j-1};\ 1 + \frac{j-1}{j})$ are Pareto optimal. This result also shows that for large values of j, we have $\lim_{j\to\infty}(1 + \frac{j-1}{j};\ \frac{j}{j-1}) = (2;\ 1)$. This means that the ratio $(2;\ 1)$ obtained by classical list scheduling algorithms for the classical MOSP problem (like the ones presented in [8]) is also Pareto optimal.

Figure 7 summarizes the inapproximation bounds presented in this section.

IV. Approximation Algorithms

We present in this section two algorithms that guarantees an approximation factor for the MOSP problem with relaxed local constraints of $(2;\ \frac{3}{2})$ and $(3;\ \frac{4}{3})$. It means that if the makespan of an organization is worsened by a factor at most of 2 (respectively 3), then the global makespan is no more than $\frac{3}{2}$ (respectively $\frac{4}{3}$) of the optimal.

Since the value for the optimal makespan is unknown, we use the dual approximation technique introduced by Hochbaum and Shmoys [15], that uses a binary search approach to estimate the value of the optimal makespan. For the sake of clarity – and without loss of generality – we rescale the length of all jobs in such a way that 1 is the length of optimal makespan.

A. Principle

The main idea of the algorithms presented in this section is to allocate optimally all the *large* jobs. This placement will be used to determine where to schedule the remaining jobs of the organizations, with respect to the relaxed local constraints. The definition of what is a *large* job depends on the value of the target global makespan. Both presented algorithms are designed on the same basic structure of job partitioning. The principle can be decomposed into five successive phases as follows:

1) all organizations are classified according to the value of their initial makespan and presence of a *large* job;
2) an algorithm is used to determine the placement for the *large* jobs;
3) some of the large jobs are used to determine where to schedule all the jobs from their owners;
4) entire organizations are migrated to machines with total load less than the optimal makespan;
5) finally, the remaining organizations (the ones with initial makespans *too small* or *too large*) are allocated and the entire schedule is produced, starting all the jobs as soon as possible in the predetermined order to remove idle times.

B. Algorithm 1: $(2;\ \frac{3}{2})$

The first approximation algorithm presented computes a schedule that is a $\frac{3}{2}$-approximation for the global makespan, while ensuring that no organization will have its makespan more than doubled if compared with its initial makespan.

Phase 1: Classify each organization into one of the following disjoint groups:

- $\mathcal{A} = \{O^{(k)} \mid C_{\max}^{(k)\ local} \leq \frac{1}{2}\}$;
- $\mathcal{B}_1 = \{O^{(k)} \mid \frac{1}{2} < C_{\max}^{(k)\ local} \leq \frac{3}{4}$ and $\nexists\ J_i^{(k)}$ such that $p_i^{(k)} > \frac{1}{2}\}$;
- $\mathcal{B}_2 = \{O^{(k)} \mid \frac{1}{2} < C_{\max}^{(k)\ local} \leq \frac{3}{4}$ and $\exists!\ J_i^{(k)}$ such that $p_i^{(k)} > \frac{1}{2}\}$;
- $\mathcal{C} = \{O^{(k)} \mid C_{\max}^{(k)\ local} > \frac{3}{4}\}$.

This classification gathers organizations that have small initial makespans ($\mathcal{A}$), that are hard to schedule with respect to the local constraints; organizations that are easy to schedule ($\mathcal{C}$), since within the global makespan of $\frac{3}{2}$ they will always fufill the local constraints; and intermediate organizations ($\mathcal{B}_1$ and $\mathcal{B}_2$) that need to be carefully scheduled. In this classification, *large* jobs are defined as jobs whose processing times $p_i^{(k)}$ are strictly larget than $\frac{1}{2}$.

Phase 2: We now assign all large jobs one per processor, from the end of the schedule under construction ($time = \frac{3}{2}$) to the beginning ($time = 0$).

Phase 3: The assignment calculated is adjusted as follows. Large jobs owned by organizations in $\mathcal{C}$ remain untouched, while each large job owned by an organization in $\mathcal{B}_2$ is replaced by all jobs of its owner (including itself).

Phase 4: Group all organizations from $\mathcal{B}_1$ and $\mathcal{B}_2$ into pairs ($O^{(i)}$ and $O^{(j)}$, where $C_{\max}^{(i)\ local} \leq C_{\max}^{(j)\ local}$) in such a way that each machine has only one pair and all jobs from $O^{(i)}$ are scheduled before the jobs from $O^{(j)}$. If $|\mathcal{B}_1| + |\mathcal{B}_2|$ is odd, then schedule the last remaining organization on their original machine at the beginning of the schedule.

Phase 5: Assign all jobs owned by organizations in $\mathcal{A}$ and the remaining jobs from the last organization $\mathcal{B}_1$ (if any) into their original machines. Then, assign all remaining jobs from organizations in $\mathcal{C}$ into the scheduling using any list scheduling algorithm. Then, compact the schedule, i.e., remove any idle time by executing early the jobs starting after an idle time.

Analysis

The Phase 2 of the algorithm presented in the previous section schedules the *large* jobs, i.e. those with $p_i^{(k)} > \frac{1}{2}$, one per processor. The number of large jobs is limited to N as shown in the following lemma:

Lemma 1: There are at most N jobs $J_i^{(k)}$ such that $p_i^{(k)} > \frac{1}{2}$.

Proof: By contradiction. Suppose that there are N organizations with $N+1$ large jobs. Since the number of large jobs is larger than the number of organizations, two large jobs with processing times $p_i^{(k)}$ and $p_j^{(l)}$ must be assigned to a same machine on the optimal schedule. Then $C_{\max} \geq p_i^{(k)} + p_j^{(l)} > 1$, which contradicts the fact that the optimal schedule must be equal to 1. ■

During Phase 4, pairs of organizations from $\mathcal{B}_2$ have all their jobs assigned to a same machine and they are scheduled one after the other. The following lemma shows why stacking two organizations does respect the α-MOSP constraints.

Lemma 2: Given two organizations $O^{(i)}$ and $O^{(j)}$, if $C_{\max}^{(i)\ local} \leq C_{\max}^{(j)\ local}$ then all jobs from these organizations can be scheduled sequentially with respect to their relaxed local constraints.

Proof: If all jobs from $O^{(i)}$ are followed by all jobs from $O^{(j)}$ on the same machine, we have $C_{\max}^{(i)} = C_{\max}^{(i)\ local}$ and

$$\begin{aligned} C_{\max}^{(j)} &= C_{\max}^{(i)\ local} + C_{\max}^{(j)\ local} \\ &\leq C_{\max}^{(j)\ local} + C_{\max}^{(j)\ local} \\ &= 2C_{\max}^{(j)\ local} \end{aligned}$$

■

At the end, organizations from $\mathcal{B}_2$ were coupled and assigned to machines with a total load greater than 1 and smaller than $\frac{3}{2}$.

Lemma 3: Jobs of organizations from $\mathcal{A}$ always can be scheduled at the beginning of the schedule before large jobs from organizations in $\mathcal{C}$ during Phase 5.

Proof: Since $|\mathcal{A}| + |\mathcal{B}_1| + |\mathcal{B}_2| \leq N$, it is sufficient to schedule organizations from $\mathcal{A}$ on the same machines where large jobs from $\mathcal{C}$ were assigned. Since a large job from $\mathcal{C}$ has processing time at most equal to 1, then the total load of the machine will be less than or equal to $\frac{1}{2} + 1 = \frac{3}{2}$. ■

Theorem 1: The schedule generated by Algorithm 1 is a $\frac{3}{2}$-approximation for the global makespan and no organization has its makespan more than doubled if compared with its initial makespan.

Proof: First, we will show that the makespan obtained by the algorithm is a $\frac{3}{2}$-approximation for the global makespan.

We start by remarking that during Phase 4, organizations from $\mathcal{B}_1$ and $\mathcal{B}_2$ are coupled and each pair is assigned to a different machine. The total load on these machines is strictly greater than $2 \cdot \frac{1}{2} = 1$ and less than or equal to $2 \cdot \frac{3}{4} = \frac{3}{2}$. If $|\mathcal{B}_1| + |\mathcal{B}_2|$ is odd, the machine with the organization that is scheduled alone has a total load bounded by $\frac{3}{4}$.

From Lemma 3, it is sufficient to schedule all jobs from $\mathcal{A}$ before the large jobs from $\mathcal{C}$. Since $C_{\max}^{(\mathcal{A})\ local} \leq \frac{1}{2}$ and $\max_{i,k} p_i^{(k)} \leq 1$, then the load on these machines are bounded by $\frac{1}{2} + 1 = \frac{3}{2}$.

Finally, after Phase 5, all remaining small jobs ($p_i^{(k)} \leq \frac{1}{2}$) are scheduled using any list scheduling algorithm. Since there always exists a machine with load < 1 available (otherwise the total work to be done would be larger than N and the optimal makespan would be larger than 1), there is always a machine in which these jobs can be scheduled. The remaining jobs are smaller than $\frac{1}{2}$, therefore the load does not exceed $1 + \frac{1}{2} = \frac{3}{2}$. This finishes the proof that the schedule generated is a $\frac{3}{2}$-approximation for the global makespan.

Furthermore, it is easy to remark that no organization will have its makespan more than doubled. Organizations from $\mathcal{A}$ will be scheduled at the beginning of the schedule and will not be delayed at all. Organizations from $\mathcal{B}_1$ and $\mathcal{B}_2$ will remain alone on their own machines or will be scheduled together. Lemma 2 guarantees that no stacked organization will be delayed. Organizations from $\mathcal{C}$ can be scheduled anywhere from $t = 0$ to $t = \frac{3}{2}$, since $2 \cdot C_{\max}^{(\mathcal{C})} \leq 2 \cdot \frac{3}{4} = \frac{3}{2}$. ■

C. Algorithm 2: $(3; \frac{4}{3})$

Using the same frame as in Algorithm 1, we define below a new algorithm which achieves a $\frac{4}{3}$-approximation for the global makespan while guaranteeing that no organization will have its local makespan more than tripled.

This algorithm also has five phases described below.

Phase 1: Classify each organization into one of the following disjoint groups:

- $\mathcal{A} = \{O^{(k)} \mid C_{\max}^{(k)\ local} \leq \frac{1}{3}\}$;
- $\mathcal{B}_1 = \{O^{(k)} \mid \frac{1}{3} < C_{\max}^{(k)\ local} \leq \frac{4}{9}$ and $\nexists\ J_i^{(k)}$ such that $p_i^{(k)} > \frac{1}{3}\}$;
- $\mathcal{B}_2 = \{O^{(k)} \mid \frac{1}{3} < C_{\max}^{(k)\ local} \leq \frac{4}{9}$ and $\exists!\ J_i^{(k)}$ such that $p_i^{(k)} > \frac{1}{3}\}$;
- $\mathcal{C} = \{O^{(k)} \mid C_{\max}^{(k)\ local} > \frac{4}{9}\}$.

Phase 2: We first pair all the large jobs using an LPT scheduling order. Since a large job have $p_i^{(k)} > \frac{1}{3}$, we have at most $2N$ large jobs owned by organizations either in $\mathcal{B}_2$ or $\mathcal{C}$. Remark that if there are x large jobs with $x \leq N$ there will be no pairing, and there will even be $N - x$ machines with no large jobs.

During this phase, we first allocate jobs from $\mathcal{B}_2$ on their own organizations, or if two jobs from $\mathcal{B}_2$ have to be scheduled together we do so on the organization with the largest index. We place paired jobs of $\mathcal{C}$ (or single jobs of $\mathcal{C}$) on the remaining organizations.

In the rest of the algorithm, we will note ($\mathcal{B}_2$,$\mathcal{C}$) the set of pairs created in this phase with one member of $\mathcal{B}_2$ and one member of $\mathcal{C}$. Similarly, we will use the sets ($\mathcal{C}$,$\mathcal{C}$) and ($\mathcal{B}_2$,$\mathcal{B}_2$) and if the number of large jobs is strictly lower than $2N$, the sets ($\mathcal{B}_2$) and ($\mathcal{C}$) will denote respectively the jobs of $\mathcal{B}_2$ and $\mathcal{C}$ which were not matched in a pair.

Phase 3: As in the previous algorithm, the assignment calculated in Phase 2 is adjusted and each large job owned by an organization in $\mathcal{B}_2$ is replaced by all the jobs of its owner (including itself). These jobs will be tied together in the rest of the algorithm and treated as a unique job.

Phase 4: Since we have at most $2N$ large jobs, the schedule generated in Phase 2 can contain machines with jobs from only one $\mathcal{B}_2$ or $\mathcal{C}$ organization, or jobs from two organizations taken in $\mathcal{B}_2$ or $\mathcal{C}$.

As long as the set ($\mathcal{B}_2$,$\mathcal{C}$) has at least two elements, we consider two such pairs. Let $O^{(i)}$ be the organization from $\mathcal{B}_2$ whose job is in the first pair from ($\mathcal{B}_2$,$\mathcal{C}$), and $O^{(j)}$ be the organization from $\mathcal{B}_2$ whose job is in the second pair. Supposing that $i > j$, we put all the jobs from $O^{(i)}$ and $O^{(j)}$ on the machine i, and the two jobs from $\mathcal{C}$ they were paired with on machine j. Jobs on machine i are ordered according to their local makespan. The sets ($\mathcal{B}_2$,$\mathcal{B}_2$), ($\mathcal{C}$,$\mathcal{C}$) and ($\mathcal{B}_2$,$\mathcal{C}$) are then updated accordingly.

This leaves us with many pairs in ($\mathcal{B}_2$,$\mathcal{B}_2$) and ($\mathcal{C}$,$\mathcal{C}$) and at most one in ($\mathcal{B}_2$,$\mathcal{C}$). Remember that the sets ($\mathcal{B}_2$) and ($\mathcal{C}$) are eventually not empty if there were less than $2N$ large jobs in Phase 2. Jobs in ($\mathcal{B}_2$) are allocated to their own organization machine.

Using a similar argument from Lemma 2 we can show that up to three organizations can be scheduled sequentially if the jobs are scheduled from the organization with smaller initial makespan to the organization with the larger makespan.

As long as there are organizations of type $\mathcal{B}_1$ which are unaffected, these organizations are distributed, allocating one to each pair in ($\mathcal{B}_2$,$\mathcal{B}_2$), and two to each single organization in ($\mathcal{B}_2$). After this stage, if there are some organizations from $\mathcal{B}_1$ which are unaffected, then all the organizations from $\mathcal{B}_2$ are either in triplets from $\mathcal{B}_1$ and $\mathcal{B}_2$ or in the last pair of ($\mathcal{B}_2$,$\mathcal{C}$).

If there were strictly less than N large jobs in Phase 2, the empty machines are filled with triplets of organizations from $\mathcal{B}_1$, as long as that is possible. After this stage, either all the jobs from organizations in $\mathcal{B}_1$ and $\mathcal{B}_2$ have been allocated somewhere, or all the machines either have a triplet from jobs in $\mathcal{B}_1$ and/or $\mathcal{B}_2$, or they have at least one large job from $\mathcal{C}$.

Phase 5: Assign all jobs owned by organizations in $\mathcal{A}$ to their original machines with an original pair of ($\mathcal{C}$,$\mathcal{C}$) from Phase 2 or a single job from ($\mathcal{C}$) if the remaining pairs in ($\mathcal{C}$,$\mathcal{C}$) are only pairs formed during Phase 4. Then, assign all remaining jobs from organizations $\mathcal{B}_1$ sequentially to any machine with less than 1 unit of workload.

Finally, assign all the remaining small jobs in $\mathcal{C}$ to the scheduling using any list scheduling algorithm. The jobs execution are ordered on each machine according to their original local makespan.

Analysis

Lemma 4: There are at most $2N$ jobs $J_i^{(k)}$ such that $p_i^{(k)} > \frac{1}{3}$

Proof: By contradiction, similar to the proof of Lemma 1. Suppose that there are N organizations with $2N + 1$ large jobs. Since the number of large jobs is larger than twice the number of organizations, three large jobs must be assigned to a same machine on the optimal schedule. Then $C_{\max} \geq 3 \cdot p_i^{(k)} > 3 \cdot \frac{1}{3} > 1$, which contradicts the fact that the optimal schedule must be equal to 1. ■

Lemma 5: When considering a pair of ($\mathcal{B}_2$,$\mathcal{C}$) formed in Phase 2, the length of the jobs from $\mathcal{C}$ is lesser than $\frac{2}{3}$, and scheduling two such jobs together is possible within the targeted global makespan.

Proof: By contradiction. Suppose that LPT scheduled the pair ($\mathcal{B}_2$,$\mathcal{C}$) together and that length of the jobs from $\mathcal{C}$ is greater than or equal to $\frac{2}{3}$. Since the makespan of organizations in $\mathcal{B}_2$ is greater than $\frac{1}{3}$, the makespan obtained on the machine where the pair were assigned is greater than $\frac{2}{3} + \frac{1}{3} = 1$, which contradicts the fact that LPT schedule the first two jobs optimally on each machine [16]. ■

Lemma 6: Scheduling all the jobs from any triplet of organizations of type $\mathcal{B}_1$ or $\mathcal{B}_2$ on a single machine is always possible within the target bounds on global makespan and degree of cooperativeness.

Proof: Since the makespan of any organization in $\mathcal{B}_1$ and $\mathcal{B}_2$ is at most $\frac{4}{9}$, scheduling a triplet of these organizations on a same machine produces a makespan of at most $3 \cdot \frac{4}{9} = \frac{4}{3}$, which respects the target bounds on global makespan.

Similarly to Lemma 2, this triplet can be scheduled sequentialy while respecting the degree of cooperativeness. Let $O^{(i)}$, $O^{(j)}$, and $O^{(k)}$ be the organizations from this triplet with $C_{\max}^{(i)\ local} \leq C_{\max}^{(j)\ local} \leq C_{\max}^{(k)\ local}$. If the organizations are scheduled from the one with smaller initial makespan to the one with larger makespan, we have:

- $C_{\max}^{(i)} = C_{\max}^{(i)\ local}$;
- $C_{\max}^{(j)} = C_{\max}^{(i)\ local} + C_{\max}^{(j)\ local} \leq 2C_{\max}^{(j)\ local}$;
- $C_{\max}^{(l)} = C_{\max}^{(i)\ local} + C_{\max}^{(j)\ local} + C_{\max}^{(l)\ local} \leq 3C_{\max}^{(l)\ local}$.

This show that all organizations have their makespan at most tripled, and respects the targeted degree of cooperativeness $\alpha = 3$. ■

Lemma 7: There are always enough machines left to assign all the organizations of type $\mathcal{A}$ to a machine where there are no jobs of type $\mathcal{B}_1$ or $\mathcal{B}_2$, and no pair of ($\mathcal{C}$,$\mathcal{C}$) formed during Phase 4.

Proof: During phase 4, pairs of ($\mathcal{C}$,$\mathcal{C}$) are formed by splitting two pairs of ($\mathcal{B}_2$,$\mathcal{C}$) and creating two new pairs by grouping the two organizations of $\mathcal{B}_1$ in one pair and the two organizations of $\mathcal{C}$ in the other. This means that the number of pairs ($\mathcal{C}$,$\mathcal{C}$) generated by Phase 4 is bounded by half of the number of organizations in $\mathcal{B}_2$ (as are pairs ($\mathcal{B}_2$,$\mathcal{B}_2$)) and these pairs ($\mathcal{C}$,$\mathcal{C}$) are actually assigned to machines originally owned by an organization in $\mathcal{B}_2$. Since $|\mathcal{A}|+|\mathcal{B}_1|+|\mathcal{B}_2| < N$, there is always a machine to assign the jobs from $\mathcal{A}$ not owned by an organization from $\mathcal{B}_1$ or $\mathcal{B}_2$. ■

Lemma 8: Whenever there is a large job of $\mathcal{C}$ scheduled on a machine and no more than 1 unit of workload, jobs from organizations $\mathcal{B}_1$ can be added to the machine while still respecting the target bounds on global makespan and degree of cooperativeness.

Proof: First, let us prove that the bound on the global makespan is respected. Individual jobs $J_i^{(k)}$ belonging to an organization $\mathcal{B}_1$ are all strictly shorter than $\frac{1}{3}$. Therefore, when adding such a job to a machine with less than 1 unit of workload, the total workload is strictly lower than $\frac{4}{3}$. Hence, when scheduling all the jobs without idle time the makespan is strictly lower than $\frac{4}{3}$.

Since the targeted degree of cooperativeness is 3, the inserted job $J_i^{(k)}$ have to complete at most at time $3C_{\max}^{(k)\ local}$. Since k is an organization of $\mathcal{B}_1$, its local makespan is more than $\frac{1}{3}$ and $3C_{\max}^{(k)\ local}$ is greater than 1. As there is a large job (*i.e.* of length more than $\frac{1}{3}$) of $\mathcal{C}$ on the machine, and this large job will be executed last with a global makespan lower than $\frac{4}{3}$, any job scheduled on the same machine will complete before time 1. In particular, job $J_i^{(k)}$ will complete before time 1, which is lower than thrice its local makespan. ■

Lemma 9: Remaining small jobs from organization $\mathcal{C}$ always have a machine to be scheduled on, while still respecting the target bounds on global makespan and degree of cooperativeness.

Proof: Within the targeted makespan of $\frac{4}{3}$, the degree of cooperativeness $\alpha = 3$ is always respected for any organization k in $\mathcal{C}$ because $C_{\max}^{(k)\ local} > \frac{4}{9} \Rightarrow 3C_{\max}^{(k)\ local} > \frac{4}{3}$. This means that the remaining small jobs from $\mathcal{C}$ can always be scheduled at the end of the schedule, after all other jobs were assigned while respecting the degree of cooperativeness.

Since there is always a machine with load less than 1, after the addition of a small job from $\mathcal{C}$ we will have a makespan smaller than $1 + \frac{1}{3} = \frac{4}{3}$ and, therefore, the target bound on the global makespan is respected. ■

Theorem 2: Algorithm 2 provides a schedule which has a global makespan lower or equal to $\frac{4}{3}$, and for which each task $J_i^(k)$ completes at most at time $3C_{\max}^{(k)\ local}$.

Proof:

- During Phase 1, the organizations are just structured in groups.
- During Phase 2, we pair large jobs to distribute them evenly and ensure that the global makespan will be lower than $\frac{4}{3}$. We prove in Lemma 4 that this pairing is possible, and at this stage the workload affected on each machine is strictly lower than 1 as proved in [16].
- During Phase 3, we add at most $\frac{1}{9}$ units of workload to pairs ($\mathcal{B}_2$,$\mathcal{C}$), and $\frac{2}{9}$ units of workload to pairs ($\mathcal{B}_2$,$\mathcal{B}_2$).
- Since the organizations of type $\mathcal{B}_2$ have a local makespan lower than $\frac{4}{9}$, and as we proved in Lemma 5 that we can bound the length of jobs from $\mathcal{C}$ paired with a job from $\mathcal{B}_2$, the transformation done in Phase 4 between two pairs of ($\mathcal{B}_2$,$\mathcal{C}$) into one pair of ($\mathcal{B}_2$,$\mathcal{B}_2$) and one pair of ($\mathcal{C}$,$\mathcal{C}$) keeps the workload of all machines under the $\frac{4}{3}$ bound.
- The triplets formed in the second part of Phase 4 can be scheduled within the targeted bounds for the global makespan and degree of cooperativeness according to Lemma 6.
- During Phase 5, jobs from organizations of type $\mathcal{A}$ are alloted to machines with a workload lower than 1, since the pairs of ($\mathcal{C}$,$\mathcal{C}$) are original pairs from Phase 2, as proved in Lemma 7. The workload on these machines is then lower or equal to $\frac{4}{3}$.
- In the second part of Phase 5, the remaining jobs from organizations $\mathcal{B}_1$ are alloted to the least utilized machines such that they will complete before thrice their local makespan as proved in Lemma 8.
- Finally, Lemma 9 states that all the remaining small jobs from organizations $\mathcal{C}$ can be scheduled within the global makespan bound.

■

Remark that this structure could be used to generate an algorithm with performance ratio of $(4; \frac{5}{4})$, or any larger values corresponding to Family 1. However, it is our conviction that these values would be of little practical interest.

V. Concluding Remarks

In this paper we studied a relaxed form of the scheduling problem known as the Multi-organization Scheduling Problem (MOSP). We investigated how limited cooperation between organizations can greatly improve the global performance of grid computing platforms. This relaxed form of MOSP is known in the literature as the α-*Cooperative Multi-Organization Scheduling Problem* (α-MOSP). It models the scheduling problem where organizations accept a limited degradation on their perceived performance in order to improve the quality of the global performance.

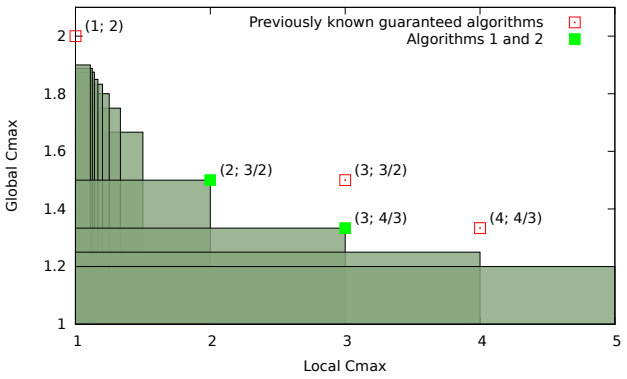


Figure 8. Graphical interpretation of the results.

We improved the previously known inapproximation bounds for α-MOSP by showing that it is actually not polynomially approximable *even if* $P = NP$. We designed two families of instances whose Pareto optimal points corroborate the presented inapproximation bounds. Then, two new algorithms with guaranteed performance to solve the α-MOSP problem were developed and analyzed. We showed that the first one achieves a $\frac{3}{2}$-approximation for the obtained global makespan, while it guarantees that no organization will have its makespan more than doubled. The second one guarantees a $\frac{4}{3}$-approximation for the global makespan, while no organization has its makespan more than tripled. We summarize in Figure 8 all these results. It evidences the improvements of the known results and shows how close the new approximation ratios are from the Pareto set.

Our future work includes two distinct lines of research. First, we are interested in the development of algorithms that produce approximation ratios that are Pareto optimal. The inapproximation analysis presented in Section III suggests that algorithms with guaranteed approximation ratios of $(2; \frac{4}{3})$ or $(3; \frac{5}{4})$ are still possible. We are also interested in the possible links between this work and Game Theory. More specifically, we are interested in the study of the Price of Anarchy on non-cooperative games and in the study of the possible relations of α-MOSP with the theory of ϵ-Nash equilibria.

References

[1] B. S. Baker, E. G. Coffman, Jr., and R. L. Rivest, "Orthogonal packings in two dimensions," *SIAM Journal on Computing*, vol. 9, no. 4, pp. 846–855, Nov. 1980.

[2] S. N. Zhuk, "Approximate algorithms to pack rectangles into several strips," *Discrete Mathematics and Applications*, vol. 16, no. 1, pp. 73–85, Jan. 2006.

[3] D. Ye, X. Han, and G. Zhang, "On-line multiple-strip packing," in *Proceedings of the 3rd International Conference on Combinatorial Optimization and Applications*, ser. LNCS, S. Berlin, Ed., vol. 5573, Jun. 2009, pp. 155–165.

[4] K. Jansen and C. Otte, "Approximation algorithms for multiple strip packing," in *Proceedings of 7th Workshop on Approximation and Online Algorithms (WAOA)*, Copenhagen, Denmark, Sep. 2009.

[5] U. Schwiegelshohn, A. Tchernykh, and R. Yahyapour, "Online scheduling in grids," in *IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, Apr. 2008, pp. 1–10.

[6] F. Pascual, K. Rzadca, and D. Trystram, "Cooperation in multi-organization scheduling," in *Euro-Par 2007 Parallel Processing*, ser. Lecture Notes in Computer Science. Springer Berlin, Aug. 2007, vol. 4641/2007, pp. 224–233.

[7] ——, "Cooperation in multi-organization scheduling," *Concurrency and Comp.: Practice & Experience*, vol. 21, no. 7, pp. 905–921, May 2009.

[8] J. Cohen, D. Cordeiro, D. Trystram, and F. Wagner, "Analysis of multi-organization scheduling algorithms," in *Euro-Par 2010 – Parallel Processing*, ser. LNCS, P. D'Ambra, M. Guarracino, and D. Talia, Eds., vol. 6272. Ischia, Italy: Springer Berlin / Heidelberg, Sep. 2010, pp. 367–379.

[9] F. Ooshita, T. Izumi, and T. Izumi, "A generalized multi-organization scheduling on unrelated parallel machines," in *International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*. Los Alamitos, CA, USA: IEEE Computer Society, Dec. 2009, pp. 26–33.

[10] E. Koutsoupias and C. Papadimitriou, "Worst-case equilibria," in *Proceedings of 16th Annual Symposium on Theoretical Aspects of Computer Science*, ser. LNCS, vol. 1563. Trier, Germany: Springer Berlin, Mar. 1999, pp. 404–413.

[11] N. Nisam, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*. Cambridge University Press, Sep. 2007.

[12] E. Even-Dar, A. Kesselman, and Y. Mansour, "Convergence time to nash equilibria," *ACM Transactions on Algorithms*, vol. 3, no. 3, p. 32, Aug. 2007.

[13] I. Caragiannis, M. Flammini, C. Kaklamanis, P. Kanellopoulos, and L. Moscardelli, "Tight bounds for selfish and greedy load balancing," in *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming*, ser. Lecture Notes in Computer Science, vol. 4051. Springer Berlin, Jun. 2006, pp. 311–322.

[14] M. Voorneveld, "Characterization of Pareto dominance," *Operations Research Letters*, vol. 31, no. 1, pp. 7–11, Jan. 2003.

[15] D. S. Hochbaum and D. B. Shmoys, "A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach," *SIAM Journal on Computing*, vol. 17, no. 3, pp. 539–551, 1988.

[16] R. L. Graham, "Bounds on multiprocessing timing anomalies," *SIAM Journal on Applied Mathematics*, vol. 17, no. 2, pp. 416–429, Mar. 1969.