Analysis of Multi-Organization Scheduling Algorithms

Johanne Cohen[1], Daniel Cordeiro[2], Denis Trystram[2], and Frédéric Wagner[2]

[1] Laboratoire d'Informatique PRiSM, Université de Versailles St-Quentin-en-Yvelines
45 avenue des États-Unis, 78035 Versailles Cedex, France
[2] LIG, Grenoble University
51 avenue Jean Kuntzmann, 38330 Montbonnot Saint-Martin, France

Abstract. In this paper we consider the problem of scheduling on computing platforms composed of several independent organizations, known as the Multi-Organization Scheduling Problem (MOSP). Each organization provides both resources and tasks and follows its own objectives. We are interested in the best way to minimize the makespan on the entire platform when the organizations behave in a selfish way.

We study the complexity of the MOSP problem with two different local objectives – makespan and average completion time – and show that MOSP is NP-Hard in both cases. We formally define a selfishness notion, by means of restrictions on the schedules. We prove that selfish behavior imposes a lower bound of 2 on the approximation ratio for the global makespan.

We present various approximation algorithms of ratio 2 which validate selfishness restrictions. These algorithms are experimentally evaluated through simulation, exhibiting good average performances.

1 Introduction

1.1 Motivation and Presentation of the Problem

The new generation of many-core machines and the now mature grid computing systems allow the creation of unprecedented massively distributed systems. In order to fully exploit such large number of processors and cores available and reach the best performances, we need sophisticated scheduling algorithms that encourage users to share their resources and, at the same time, that respect each user's own interests.

Many of these new computing systems are composed of organizations that own and manage clusters of computers. A user of such systems submits his/her jobs to a scheduler system that can choose any available machine in any of these clusters. However, each organization that shares its resources aims to take maximum advantage of its own hardware. In order to improve cooperation between the organizations, local jobs should be prioritized.

To find an efficient schedule for the jobs using the available machines is a crucial problem. Although each user submits jobs locally in his/her own organization, it is necessary to optimize the allocation of the jobs for the whole

P. D'Ambra, M. Guarracino, and D. Talia (Eds.): Euro-Par 2010, Part II, LNCS 6272, pp. 367–379, 2010.

platform in order to achieve good performance. The global performance and the performance perceived by the users will depend on how the scheduler allocates resources among all available processors to execute each job.

1.2 Related Work

From the classical scheduling theory, the problem of scheduling parallel jobs is related to the Strip packing [1]. It corresponds to pack a set of rectangles (without rotations and overlaps) into a strip of machines in order to minimize the height used. Then, this problem was extended to the case where the rectangles were packed into a finite number of strips [16, 15]. More recently, an asymptotic $(1+\epsilon)$-approximation AFPTAS with additive constant $O(1)$ and with running-time polynomial in n and in $1/\epsilon$ was presented in [8].

Schwiegelshohn, Tchernykh, and Yahyapour [14] studied a very similar problem, where the jobs can be scheduled in non-contiguous processors. Their algorithm is a 3-approximation for the maximum completion time (makespan) if all jobs are known in advance, and 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. [12, 13] and studies 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 was presented with the target of minimizing the makespan, resulting in a centralized 3-approximation algorithm. This problem was then extended for relaxed local objectives in [11].

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 [9] 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 [10] 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 [4], characterization of the worst-case equilibria [3], etc. We are not targeting here at such game theoretical approaches.

1.3 Contributions and Road Map

As suggested in the previous section, the problem of scheduling in multi-organization clusters has been studied from several different points of view. In this paper, we propose a theoretical analysis of the problem using classical combinatorial optimization approaches.

Our main contribution is the extension and analysis of the problem for the case in which sequential jobs are submitted by *selfish organizations* that can handle different local objectives (namely, makespan and average completion times).

We introduce new restrictions to the schedule that take into account the notion of *selfish organizations*, i.e., organizations that refuse to cooperate if their objectives could be improved just by executing earlier one of their jobs in one of their own machines. The formal description of the problem and the notations used in this paper are described in Section 2. The Section 3 shows that any algorithm respecting our new selfishness restrictions can not achieve approximation ratios better than 2 and that both problems are intractable. New heuristics for solving the problem are presented in Section 4. Simulation experiments, discussed in Section 5, show the good results obtained by our algorithms in average.

2 Problem Description and Notations

In this paper, we are interested in the scheduling problem in which different *organizations* own a physical cluster of identical machines that are interconnected. They 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 try to minimize their own objectives – either the makespan or the average 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 can refuse to cooperate if in the final schedule one of its migrated jobs could be executed earlier in one of the machines owned by the organization.

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 $m^{(k)}$ identical machines available that 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[1]. 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 $C_{\max}^{(k)} = \max_{1 \leq i \leq n^{(k)}} (C_i^{(k)})$ and its sum of completion times as $\sum C_i^{(k)}$. The global makespan for the entire grid computing system is defined as $C_{\max} = \max_{1 \leq k \leq N} (C_{\max}^{(k)})$.

2.1 Local Constraint

The *Multi-Organization Scheduling Problem*, as first described in [12] consists in minimizing the global makespan ($C_{\max}$) with an additional *local constraint*: at the end, no organization can have its makespan increased if compared with the makespan that the organization could have by scheduling the jobs in its

[1] All machines are identical, i.e., every job will be executed at the same speed independently of the chosen machine.

own machines ($C_{\max}^{(k)\ local}$). More formally, we call $\mathbf{MOSP}(C_{\max})$ the following optimization problem:

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

In this work, we also study the case where all organizations are interested locally in minimizing their average completion time while minimizing the global makespan. As in MOSP($C_{\max}$), each organization imposes that the sum of completion times of its jobs can not be increased if compared with what the organization could have obtained using only its own machines ($\sum C_i^{(k)\ local}$). We denote this problem $\mathbf{MOSP}(\sum C_i)$ and the goal of this optimization problem is to:

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

2.2 Selfishness

In both MOSP($C_{\max}$) and MOSP($\sum C_i$), while the global schedule might be computed by a central entity, the organizations keep control on the way they execute the jobs in the end. This property 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.

In order to prevent such behavior, we define a new restriction on the schedule, called *selfishness restriction*. The idea is that, in any schedule respecting this restriction, no single organization can improve its local schedule by cheating.

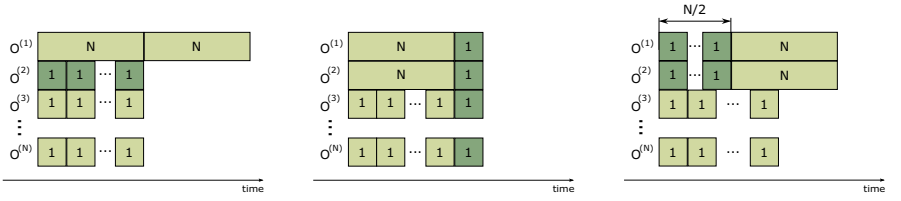
Given a fixed schedule, let $J_f^{(l)}$ be the first foreign job scheduled to be executed in $O^{(k)}$ (or the first idle time if $O^{(k)}$ has no foreign job) and $J_i^{(k)}$ any job belonging to $O^{(k)}$. Then, the *selfishness restriction* forbids any schedule where $C_f^{(l)} < C_i^{(k)} - p_i^{(k)}$. In other words, $O^{(k)}$ 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.

3 Complexity Analysis

3.1 Lower Bounds

Pascual et al. [12] showed with an instance having two organizations and two machines per organization that every algorithm that solves MOSP (for rigid, parallel jobs and $C_{\max}$ as local objectives) has at least a $\frac{3}{2}$ approximation ratio when compared to the optimal makespan that could be obtained *without the local constraints*. We show that the same bound applies asymptotically even with a larger number of organizations.

Take the instance depicted in Figure 1a. $O^{(1)}$ initially has two jobs of size N and all the others initially have N jobs of size 1. All organizations contribute



(a) Initial instance – $C_{\max} = 2N$

(b) Global optimum without constraints – $C_{\max} = N + 1$

(c) Optimum with MOSP constraints – $C_{\max} = \frac{3N}{2}$

Fig. 1. Ratio between global optimum makespan and the optimum makespan that can be obtained for both MOSP($C_{\max}$) and MOSP($\sum C_i$). Jobs owned by organization $O^{(2)}$ are highlighted.

only with 1 machine each. The optimal makespan for this instance is $N + 1$ (Figure 1b), nevertheless it delays jobs from $O^{(2)}$ and, as consequence, does not respect MOSP's local constraints. The best possible makespan that respects the local constraints (whenever the local objective is the makespan or the average completion time) is $\frac{3N}{2}$, as shown in Figure 1c.

3.2 Selfishness and Lower Bounds

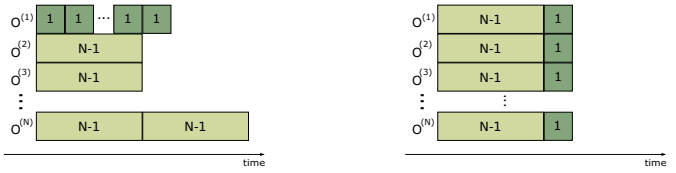
Although all organizations will likely cooperate with each other to achieve the best global makespan possible, their selfish behavior will certainly impact the quality of the best attainable global makespan.

We study here the impact of new selfishness restrictions on the quality of the achievable schedules. We show that these restrictions impact MOSP($C_{\max}$) and MOSP($\sum C_i$) as compared with unrestricted schedules and, moreover, that MOSP($C_{\max}$) with selfishness restrictions suffers from limited performances as compared to MOSP($C_{\max}$) with local constraints.

Proposition 1. *Any approximation algorithm for both* MOSP($C_{\max}$) *and* MOSP($\sum C_i$) *has ratio greater than or equal to* 2 *regarding the optimal makespan* without constraints *if all organizations behave selfishly.*

Proof. We prove this result by using the example described in Figure 1. It is clear from Figure 1b that an optimal solution for a schedule without local constraints can be achieved in $N + 1$. However, with added selfishness restrictions, Figure 1a (with a makespan of $2N$) represents the only valid schedule possible. We can, therefore, conclude that local constraints combined with selfishness restrictions imply that no algorithm can provide an approximation ratio of 2 when compared with the problem without constraints. □

Proposition 1 gives a ratio regarding the optimal makespan without the local constraints imposed by MOSP. We can show that the same approximation ratio of 2 also applies for MOSP($C_{\max}$) regarding the optimal makespan even if MOSP constraints are respected.



(a) Initial instance – $C_{\max} = 2N - 2$

(b) Global optimum with MOSP constraints – $C_{\max} = N$

Fig. 2. Ratio between global optimum makespan with MOSP constraints and the makespan that can be obtained by MOSP($C_{\max}$) with selfish organizations

Proposition 2. *Any approximation algorithm for* MOSP($C_{\max}$) *has ratio greater than or equal to* $2 - \frac{2}{N}$ *regarding the optimal makespan* with local constraints *if all organizations behave selfishly.*

Proof. Take the instance depicted in Figure 2a. $O^{(1)}$ initially has N jobs of size 1 and $O^{(N)}$ has two jobs of size N. The optimal solution that respects MOSP local constraints is given in Figure 2b and have $C_{\max}$ equal to N. Nevertheless, the best solution that respects the selfishness restrictions is the initial instance with a $C_{\max}$ equal to $2N-2$. So, the ratio of the optimal solution with the selfishness restrictions to the optimal solution with MOSP constraints is $2 - \frac{2}{N}$. □

3.3 Computational Complexity

This section studies how hard it is to find optimal solutions for MOSP even for the simpler case in which all organizations contribute only with one machine and two jobs. We consider the decision version of the MOSP defined as follows:

Instance: a set of N organizations (for $1 \leq k \leq N$, organization $O^{(k)}$ has $n^{(k)}$ jobs, $m^{(k)}$ identical machines, and makespan as the local objective) and an integer ℓ.

Question: Does there exist a schedule with a makespan less than ℓ ?

Theorem 1. MOSP($C_{\max}$) *is strongly NP-complete.*

Proof. It is straightforward to see that MOSP($C_{\max}$) $\in NP$. Our proof is based on a reduction from the well-known 3-Partition problem [5]:

Instance: a bound $B \in Z^+$ and a finite set A of $3m$ integers $\{a_1, \ldots, a_{3m}\}$, such that every element of A is strictly between $B/4$ and $B/2$ and such that $\sum_{i=1}^{3m} a_i = mB$.

Question: can A be partitioned into m disjoint sets $A_1, A_2, \ldots, A_m$ such that, for all $1 \leq i \leq m$, $\sum_{a \in A_i} a = B$ and A_i is composed of exactly three elements?

Given an instance of 3-Partition, we construct an instance of MOSP where, for $1 \leq k \leq 3m$, organization $O^{(k)}$ initially has two jobs $J_1^{(k)}$ and $J_2^{(k)}$ with

$p_1^{(k)} = (m+1)B + 7$ and $p_2^{(k)} = (m+1)a_k + 1$, and all other organizations have two jobs with processing time equal to 2. We then set ℓ to be equal to $(m+1)B + 7$. Figure 3 depicts the described instance. This construction is performed in polynomial time. Now, we prove that A can be split into m disjoint subsets $A_1, \ldots, A_m$, each one summing up to B, if and only if this instance of MOSP has a solution with $C_{max} \leq (m+1)B + 7$.

Assume that $A = \{a_1, \ldots, a_{3m}\}$ can be partitioned into m disjoint subsets $A_1, \ldots, A_m$, each one summing up to B. In this case, we can build an optimal schedule for the instance as follows:

- for $1 \leq k \leq 3m$, $J_1^{(k)}$ is scheduled on machine k;
- for $3m+1 \leq k \leq 4m$, $J_1^{(k)}$ and $J_2^{(k)}$ are scheduled on machine k;
- for $1 \leq i \leq m$, let $A_i = \{a_{i_1}, a_{i_2}, a_{i_3}\} \subseteq A$. The jobs $J_2^{(a_{i_1})}$, $J_2^{(a_{i_2})}$ and $J_2^{(a_{i_3})}$ are scheduled on machine $3m+i$.

So, the global $C_{\max}$ is $(m+1)B + 7$ and the local constraints are respected.

Conversely, assume that MOSP has a solution with $C_{max} \leq (m+1)B + 7$. The total work (W) of the jobs that must be executed is $W = 3m((m+1)B + 7) + 2 \cdot 2m + (m+1)\sum_{i=1}^{3m} a_i + 3m = 4m(m+1)B + 7$. Since we have exactly $4m$ organizations, the solution must be the optimal solution and there are no idle times in the scheduling. Moreover, $3m$ machines must execute only one job of size $(m+1)B + 7$. W.l.o.g, we can consider that for $3m+1 \leq k \leq 4m$, machine k performs jobs of size less than $(m+1)B + 7$.

To prove our proposition, we first show two lemmas:

Lemma 1. *For all $3m+1 \leq k \leq 4m$, at most four jobs of size not equal to 2 can be scheduled on machine k if $C_{\max}^{(k)} \leq (m+1)B + 7$.*

Proof. It is enough to notice that all jobs of size not equal to 2 are greater than $(m+1)B/4+1$, that $C_{\max}$ must be equal to $(m+1)B+7$ and that $m+1 > 3$. □

Lemma 2. *For all $3m+1 \leq k \leq 4m$, exactly two jobs of size 2 are scheduled on each machine k if $C_{\max}^{(k)} \leq (m+1)B + 7$.*

Proof. We prove this lemma by contradiction. Assume that there exists a machine k such that at most one job of size 2 is scheduled on it. So, by definition of the size of jobs, all jobs scheduled in machine k have a size greater than $(m+1)B/4 + 1$. By consequence of Lemma 1, since at most four jobs can be scheduled on machine k, the total work on this machine is $(m+1)B + y + 2$ where $y \leq 4$. This fact is in contradiction with the facts that there does not exist idle processing time and that $K = (m+1)B + 7$. □

Now, we construct m disjoint subsets $A_1, A_2, \ldots, A_m$ of A as follows: for all $1 \leq i \leq m$, a_j is in A_i if the job with size $(m+1)a_j + 1$ is scheduled on machine $3m+i$. Note that all elements of A belong to one and only one set in $\{A_1, \ldots, A_m\}$. We prove that A is a partition with desired properties. We focus

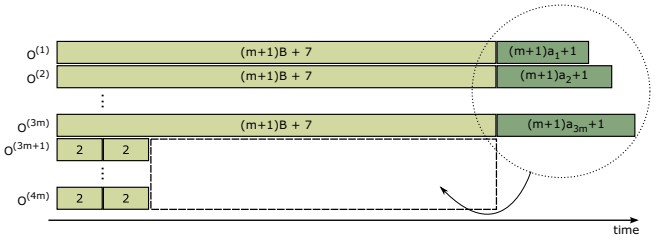


Fig. 3. Reduction of MOSP($C_{\max}$) from 3-PARTITION

on a fixed element A_i. By definition of A_i we have that

$$4 + \sum_{a_j \in A_i} ((m+1)a_j + 1) = (m+1)B + 7 \Rightarrow \sum_{a_j \in A_i} ((m+1)a_j + 1 = (m+1)B + 3$$

Since $m+1 > 3$, we have $\sum_{a_j \in A_i}(m+1)a_j = (m+1)B$. Thus, we can deduce that A_i is composed of exactly three elements and $\sum_{a \in A_i} a = B$. □

We continue by showing that even if all organizations are interested locally in the average completion time, the problem is still NP-complete. We prove NP-completeness of the MOSP($\sum C_i$) problem (having a formulation similar to the MOSP($C_{\max}$) decision problem) using a reduction from the PARTITION problem. The idea here is similar to the one used in the previous reduction, but the $\sum C_i$ constraints heavily restrict the allowed movements of jobs when compared to the $C_{\max}$ constraints.

Theorem 2. MOSP($\sum C_i$) *is NP-complete.*

Proof. First, note that it is straightforward to see that MOSP($\sum C_i$) $\in NP$. We use the PARTITION [5] problem to prove this theorem.

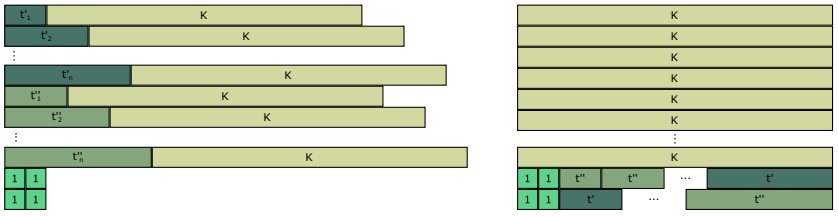
Instance: a set of n integers $s_1, s_2, \ldots, s_n$.

Question: does there exist a subset $J \subseteq I = \{1, \ldots, n\}$ such that

$$\sum_{i \in J} s_i = \sum_{i \in I \setminus J} s_i \ ?$$

Consider an integer $M > \sum_i s_i$. Given an instance of the PARTITION problem, we construct an instance of MOSP($\sum C_i$) problem, as depicted in Figure 4a. There are $N = 2n + 2$ organizations having two jobs each. The organizations $O^{(2n+1)}$ and $O^{(2n+2)}$ have two jobs with processing time 1.

Each integer s_i from the PARTITION problem corresponds to a pair of jobs t'_i and t''_i, with processing time equal to $2^i M$ and $2^i M + s_i$ respectively. We set $J_1^{(k)} = t'_k$, for all $1 \leq k \leq n$ and $J_1^{(k)} = t''_{k-n}$, for all $n+1 \leq k \leq 2n$. We set K to $\frac{W}{N} = \frac{\sum_i t'_i + \sum_i t''_i + 4}{2}$. To complete the construction, for any k, $1 \leq k \leq 2n$, the organization $O^{(k)}$ has also a job $J_2^{(k)}$ with processing time equal to K. We set



(a) Initial instance (b) Optimum

Fig. 4. Reduction of MOSP($\sum C_i$) from PARTITION

K to ℓ. This construction is performed in polynomial time and we prove that it is a reduction.

First, assume that $\{s_1, s_2, \ldots, s_n\}$ is partitioned into 2 disjoint sets S_1, S_2 with the desired properties. We construct a valid schedule with optimal global makespan for MOSP($\sum C_i$). For all s_i, if $i \in J$, we schedule job t'_i in organization $O^{(N)}$ and job t''_i in organization $O^{(N-1)}$. Otherwise, we schedule t'_i in $O^{(N-1)}$ and t''_i in $O^{(N)}$.

The problem constraints impose that organizations $O^{(N-1)}$ and $O^{(N)}$ will first schedule their own jobs (two jobs of size 1). The remaining jobs will be scheduled in non-decreasing time, using the Shortest Processing Time first (SPT) rule. This schedule respects MOSP's constraints of not increasing the organization's average completion time because each job is being delayed by at most its own size (by construction, the sum of all jobs scheduled before the job being scheduled is smaller than the size of the job). $C_{\max}^{(N-1)}$ will be equal to $2 + \sum_i 2^i M + \sum_{i \in J} s_i$. Since J is a partition, $C_{\max}^{(N-1)}$ is exactly equal to $C_{\max}^{(N)} = 2 + \sum_i 2^i M + \sum_{i \in I \setminus J} s_i$. Also, $C_{\max}^{(N)} = C_{\max}^{(N-1)} = K$, which gives us the theoretical lower bound for $C_{\max}$.

Second, assume MOSP($\sum C_i$) has a solution with $C_{\max} \leq K$. We prove that $\{s_1, s_2, \ldots, s_n\}$ is partitioned into 2 disjoint sets S_1, S_2 with the desired properties. This solution of MOSP($\sum C_i$) has the structure drawn in Figure 4b. To achieve a $C_{\max}$ equal to K, the scheduler must keep all jobs that have size exactly equal to K in their initial organizations. Moreover all jobs of size 1 must also remain in their initial organizations, otherwise these jobs would be delayed.

The remaining jobs (all t'_i and t''_i jobs) must be scheduled either in organizations $O^{(N-1)}$ or $O^{(N)}$. Each processor must execute a total work of $\frac{2K-4}{2} = \frac{2\sum_i 2^i M + \sum_i s_i}{2} = \sum_i 2^i M + \frac{\sum_i s_i}{2}$ to achieve a makespan equal to K.

Let $J \subseteq I = \{1, \ldots, n\}$ such that $i \in J$ if t''_i was scheduled on organization $O^{(N-1)}$. $O^{(N-1)}$ execute a total work of $W^{(N-1)} = \sum_i 2^i M + \sum_{i \in J} s_i$, that must be equal to the total work of $O^{(N)}$, $W^{(N)} = \sum_i 2^i M + \sum_{i \in I \setminus J} s_i$. Since $\sum_i s_i < M$, we have $W^{(N-1)} \equiv \sum_{i \in J} s_i \pmod M$ and $W^{(N)} \equiv \sum_{i \in I \setminus J} \pmod M$. This means that $W^{(N-1)} = W^{(N)} \implies (W^{(N-1)} \bmod M) = (W^{(N)} \bmod M) \implies \sum_{i \in J} s_i = \sum_{i \in I \setminus J} s_i$. If MOSP($\sum C_i$) has a solution with $C_{\max} \leq K$, then set J is a solution for PARTITION. □

4 Algorithms

In this section, we present three different heuristics to solve MOSP($C_{\max}$) and MOSP($\sum C_i$). All algorithms present the additional property of respecting selfishness restrictions.

4.1 Iterative Load Balancing Algorithm

The Iterative Load Balancing Algorithm (ILBA) [13] is a heuristic that redistributes the load from the most loaded organizations.

The idea is to incrementally rebalance the load without delaying any job. First the less loaded organizations are rebalanced. Then, one-by-one, each organization has its load rebalanced.

The heuristic works as follows. First, each organization schedules its own jobs locally and the organizations are enumerated by non-decreasing makespans, i.e. $C_{\max}^{(1)} \leq C_{\max}^{(2)} \leq \ldots \leq C_{\max}^{(N)}$. For $k = 2$ until N, jobs from $O^{(k)}$ are rescheduled sequentially, and assigned to the less loaded of organizations $O^{(1)} \ldots O^{(k)}$.

Each job is rescheduled by ILBA either earlier or at the same time that the job was scheduled before the migration. In other words, no job is delayed by ILBA, which guarantees that the local constraint is respected for MOSP($C_{\max}$) and MOSP($\sum C_i$).

4.2 LPT-LPT and SPT-LPT Heuristics

We developed and evaluated (see Section 5) two new heuristics based on the classical LPT (Longest Processing Time First [6]) and SPT (Smallest Processing Time First [2]) algorithms for solving MOSP($C_{\max}$) and MOSP($\sum C_i$), respectively. Both heuristics work in two phases. During the first phase, all organizations minimize their own local objectives. Each organization starts applying LPT for its own jobs if the organization is interested in minimizing its own makespan, or starts applying SPT if the organization is interested in its own average completion time.

The second phase is when all organizations cooperatively minimize the makespan of the entire grid computing system without worsening any local objective. This phase works as follows: each time an organization becomes idle, i.e., it finishes the execution of all jobs assigned to it, the longest job that does not have started yet is migrated and executed by the idle organization. This greedy algorithm works like a global LPT, always choosing the longest job yet to be executed among jobs from all organizations.

4.3 Analysis

ILBA, LPT-LPT and SPT-LPT do not delay any of the jobs when compared to the initial local schedule. During the rebalancing phase, all jobs either remain in their original organization or are migrated to an organization that became idle at a preceding time. The implications are:

- the *selfishness* restriction is respected – if a job is migrated, it will start before the completion time of the last job of the initial organization;
- if organizations' local objective is to minimize the makespan, migrating a job to a previous moment in time will decrease the job's completion time and, as consequence, it will not increase the initial makespan of the organization;
- if organizations' local objective is to minimize the average completion time, migrating a job from the initial organization to another that became idle at a previous moment in time will decrease the completion time of all jobs from the initial organization and of the job being migrated. This means that the $\sum C_i$ of the jobs from the initial organization is always decreased;
- the rebalancing phase of all three algorithms works as the list scheduling algorithms. Graham's classical approximation ratio $2 - \frac{1}{N}$ of list scheduling algorithms [6] holds for all of them.

We recall from Section 3.2 that no algorithm respecting selfishness restrictions can achieve an approximation ratio for $\mathrm{MOSP}(C_{\max})$ better than 2. Since all our algorithms reach an approximation ratio of 2, no further enhancements are possible without removing selfishness restrictions.

5 Experiments

We conducted a series of simulations comparing ILBA, LPT-LPT, and SPT-LPT under various experimental settings. The workload was randomly generated with parameters matching the typical environment found in academic grid computing systems [13]. We evaluated the algorithms with instances containing a random number of machines, organizations and jobs with different sizes. 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 [7].

We are interested in the improvement of the global $C_{\max}$ provided by the different algorithms. The results are evaluated with comparison to the $C_{\max}$ obtained by the algorithms with the well-known theoretical lower bound for the scheduling problem without constraints $LB = \max(\sum_{i,k} \frac{p_i^{(k)}}{m^{(k)}}, p_{\max})$.

Our main conclusion is that, despite the fact that the selfishness restrictions are respected by all heuristics, ILBA and LPT-LPT obtained near optimal results for most cases. This is not unusual, since it follows the patterns of experimental behavior of standard list scheduling algorithms, in which it is easy to obtain a near optimal schedule when the number of tasks grows large. SPT-LPT produces worse results due to the effect of applying SPT locally.

However, in some particular cases, in which the number of jobs is not much larger than the number of machines available, the experiments yield more interesting results. Figure 5 shows the histogram of a representative instance of such a particular case. The histograms show the frequency of the ratio $C_{\max}$ obtained to the lower bound over 5000 different instances with 20 organizations and 100 jobs for ILBA, LPT-LPT and SPT-LPT. Similar results have been obtained for

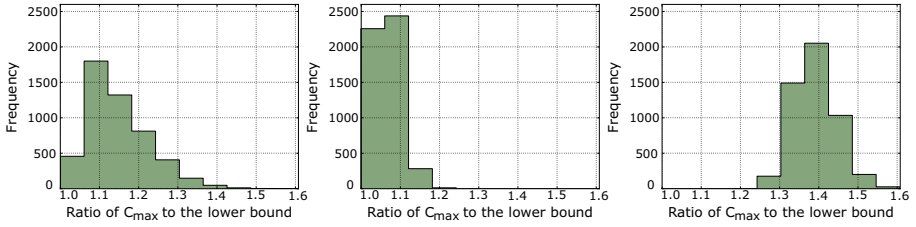


Fig. 5. Frequency of results obtained by ILBA, LPT-LPT, and SPT-LPT when the results are not always near optimal

many different sets of parameters. LPT-LPT outperforms ILBA (and SPT-LPT) for most instances and its average ratio to the lower bound is less than 1.3.

6 Concluding Remarks

In this paper, we have investigated the scheduling on multi-organization platforms. We presented the MOSP($C_{\max}$) problem from the literature and extended it to a new related problem MOSP($\sum C_i$) with another local objective. In each case we studied how to improve the global makespan while guaranteeing that no organization will worsen its own results.

We showed first that both versions MOSP($C_{\max}$) and MOSP($\sum C_i$) of the problem are NP-hard. Furthermore, we introduced the concept of *selfishness* in these problems which corresponds to additional scheduling restrictions designed to reduce the incentive for the organizations to cheat locally and disrupt the global schedule. We proved that any algorithm respecting selfishness restrictions can not achieve a better approximation ratio than 2 for MOSP($C_{\max}$).

Two new scheduling algorithms were proposed, namely LPT-LPT and SPT-LPT, in addition to ILBA from the literature. All these algorithms are list scheduling, and thus achieve a 2-approximation. We provided an in-depth analysis of these algorithms, showing that all of them respect the selfishness restrictions.

Finally, all these algorithms were implemented and analysed through experimental simulations. The results show that our new LPT-LPT outperforms ILBA and that all algorithms exhibit near optimal performances when the number of jobs becomes large.

Future research directions will be more focused on game theory. We intend to study schedules in the case where several organizations secretly cooperate to cheat the central authority.

References

1. Baker, B.S., Coffman Jr., E.G., Rivest, R.L.: Orthogonal packings in two dimensions. SIAM Journal on Computing 9(4), 846–855 (1980)
2. Bruno, J.L., Coffman Jr., E.G., Sethi, R.: Scheduling independent tasks to reduce mean finishing time. Communications of the ACM 17(7), 382–387 (1974)

3. Caragiannis, I., Flammini, M., Kaklamanis, C., Kanellopoulos, P., Moscardelli, L.: Tight bounds for selfish and greedy load balancing. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4051, pp. 311–322. Springer, Heidelberg (2006)
4. Even-Dar, E., Kesselman, A., Mansour, Y.: Convergence time to nash equilibria. ACM Transactions on Algorithms 3(3), 32 (2007)
5. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, New York (January 1979)
6. Graham, R.L.: Bounds on multiprocessing timing anomalies. SIAM Journal on Applied Mathematics 17(2), 416–429 (1969)
7. Iosup, A., Dumitrescu, C., Epema, D., Li, H., Wolters, L.: How are real grids used? The analysis of four grid traces and its implications. In: 7th IEEE/ACM International Conference on Grid Computing, pp. 262–269 (September 2006)
8. Jansen, K., Otte, C.: Approximation algorithms for multiple strip packing. In: Bampis, E., Jansen, K. (eds.) WAOA 2009. LNCS, vol. 5893, pp. 37–48. Springer, Heidelberg (2010)
9. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibria. In: Meinel, C., Tison, S. (eds.) STACS 1999. LNCS, vol. 1563, pp. 404–413. Springer, Heidelberg (1999)
10. Nisam, N., Roughgarden, T., Tardos, E., Vazirani, V.V.: Algorithmic Game Theory. Cambridge University Press, Cambridge (September 2007)
11. Ooshita, F., Izumi, T., Izumi, T.: A generalized multi-organization scheduling on unrelated parallel machines. In: International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), pp. 26–33. IEEE Computer Society, Los Alamitos (December 2009)
12. Pascual, F., Rzadca, K., Trystram, D.: Cooperation in multi-organization scheduling. In: Kermarrec, A.-M., Bougé, L., Priol, T. (eds.) Euro-Par 2007. LNCS, vol. 4641, pp. 224–233. Springer, Heidelberg (August 2007)
13. Pascual, F., Rzadca, K., Trystram, D.: Cooperation in multi-organization scheduling. Concurrency and Comp.: Practice & Experience 21(7), 905–921 (2009)
14. Schwiegelshohn, U., Tchernykh, A., Yahyapour, R.: Online scheduling in grids. In: IEEE International Symposium on Parallel and Distributed Processing (IPDPS), pp. 1–10 (April 2008)
15. Ye, D., Han, X., Zhang, G.: On-line multiple-strip packing. In: Berlin, S. (ed.) Proceedings of the 3rd International Conference on Combinatorial Optimization and Applications, June 2009. LNCS, vol. 5573, pp. 155–165. Springer, Heidelberg (2009)
16. Zhuk, S.N.: Approximate algorithms to pack rectangles into several strips. Discrete Mathematics and Applications 16(1), 73–85 (2006)