Energy-Aware Multi-Organization Scheduling Problem*

Johanne Cohen[1], Daniel Cordeiro[2], and Pedro Luis F. Raphael[2]

[1] Laboratoire de Recherche en Informatique (LRI, UMR 8623),
Université Paris-Sud, Bât 650 Ada Lovelace, 91405 Orsay, France
Johanne.Cohen@lri.fr
[2] Department of Computer Science,
University of São Paulo, Rua do Matão, 1010; 05508-090 São Paulo/SP, Brazil
{danielc,plfr}@ime.usp.br

Abstract. Scheduling algorithms for shared platforms such as grids and clouds granted users of different organizations access to powerful resources and may improve machine utilization; however, this can also increase operational costs of less-loaded organizations.

We consider energy as a resource, where the objective is to optimize the total energy consumption without increasing the energy spent by a *selfish organization*. We model the problem as a energy-aware variant of the Multi-Organization Scheduling Problem that we call MOSP-ENERGY.

We show that the clairvoyant problem with variable speed processors and jobs with release dates and deadlines is NP-hard and also that being selfish can cause solutions at most $m^{\alpha-1}$ far from the optimal, where m is the number of machines and $\alpha > 1$ is a constant. Finally, we present efficient heuristics for scenarios with all jobs ready from the beginning.

1 Introduction

Cooperative computational platforms such as grid computing or community clouds are typically organized as a federated system where users and computational resources, belonging to different organizations — *i.e.*, different administrative domains — share resources and exchange jobs with each other, in order to simultaneously maximize the profits of the collectivity and their own interests. Those platforms create novel research and business possibilities that, in turn, require ever more computational power. Examples of such organizations are research laboratories, universities or company departments.

Current distributed systems and their underlying algorithms allow an efficient redistribution of the jobs over the available resources, improving the overall utilization of the platform. Specialized algorithms for cooperative computing are capable of incite the creation of these platforms by guaranteeing that no organization will worsen its own results (in terms of performance) by sharing its resources with the others, even when the other behave in a selfish way.

* This work was partially funded by the São Paulo Research Foundation (FAPESP #2012/03778-0).

F. Silva et al. (Eds.): Euro-Par 2014, LNCS 8632, pp. 186–197, 2014.

The participation on such communities can have a side-effect that is often neglected by its users: the unpredictable increase of the operational costs for the organization. Less loaded organizations could save energy by putting its machines on stand-by, turning them off, or even decreasing the speed of the processors for non-priority jobs. The co-existence of these jobs with jobs migrated from other organizations can make this practice unfeasible.

It is crucial to optimize the allocation of the jobs for the whole platform in order to achieve good system performances. Moreover, it is important to do that in such a way that no organization will be harmed by sharing its own resources. The goal of this work is to study this problem considering energy costs also as a kind of resource that should be exchanged between the participants.

1.1 Related Work

The evolution of the processors technology has been driven by the demand of increased performance and reduced sizes. These demands resulted on chips with high power density and temperatures. On large scale server farms, energy-efficiency became an important issue because of the energy costs. Furthermore, part of this energy is converted into heat, which degrades processor's performance and reliability. Technologies as Intel's "Turbo Boost" or AMD's "Power-Now" were developed to offer speed-scaling capabilities, that allow the system to set the speed of the processors in order to control energy consumption.

The *Dynamic Speed Scaling* scheduling model was first studied by Yao, Demers and Shenker [9]. They considered a problem where n jobs with release dates r_i, deadlines d_i and processing volumes w_i, must be scheduled in a variable-speed processor with the objective of minimizing the energy consumption on that processor. The energy consumption is given by the integral over time of the power function $P(s(t)) = s(t)^\alpha$, where $s(t)$ is the speed in which the processor is running on time t and $\alpha > 1$ is a constant real number that depends on the technical characteristics of the processor — usually $\alpha \in [2, 3]$. There are two assumptions to simplify the model: the processor spectrum of speeds is continuous and can be any real number between $0 \leq s \leq +\infty$.

They have proposed an optimal greedy algorithm for the problem, known as the *YDS algorithm*. It iteractively finds the *maximum density interval*, that is, the time interval $[t, t']$ such that the sum of the processing volumes of the jobs completely inside that interval, divided by the length of the interval, is maximum. By the convexity of the power function, this value gives the optimal speed on that interval (in the sense that no other feasible schedule can use less power on that interval.) The jobs in the interval are then scheduled using the Earliest Deadline First policy at this speed, jobs partially in the interval have their release dates and deadlines adjusted.

Albers et al. [1,2] studied the problem with m variable-speed processors with and without preemption and job migration. When migration is not allowed, the problem is NP-Complete; otherwise there is a polynomial algorithm to find the optimal solution. They also proved that, if the jobs have *agreeable deadlines* (*i.e.*, given two jobs, if $r_1 \leq r_2$ then $d_1 \leq d_2$), the problem can be optimally solved in

polynomial-time by distributing the jobs in a round-robin fashion, prioritizing jobs with smaller release dates.

Scheduling on cooperative platforms were first studied by Pascual et al. [5,8]. They proposed the Multi-Organization Scheduling Problem (MOSP). In their model, independent organizations, sharing resources on a grid-like fashion, have a local performance objective 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 individualists organizations has been broaden by Cohen et al. [4]. Studying workloads of bag-of-tasks jobs, they have analyzed situations where selfish organizations could change the schedule of the jobs assigned to its own machines and proposed algorithms that avoid schedules where the devised global schedule could be changed by re-inserting local jobs earlier. When all organizations behave selfishly, any approximation algorithm has a ratio greater than or equal to $(2 - N/2)$ regarding the optimal makespan with local constraints and presented several 2-approximation algorithms for the global makespan that always respect the selfishness restriction. They have also analyzed the decentralization of the decision making using Algorithmic Game Theory [3].

1.2 Contributions and Outline of this Paper

Scheduling algorithms for modern cooperative platforms composed of resources shared between independent participants granted its user access to powerful resources and improved the utilization of machines that were, most of the time, idle. With the increasing need for more computational power, the energy consumption on these machine also became an issue.

We modeled the problem as a Multi-Organization Scheduling Problem (MOSP) with respect to the system total energy consumption. We have multiple organizations, each one with a processor that can operate at variable speed (as in classic Dynamic Speed Scaling problems), and its own set of jobs. The goal is to find a global schedule, migrating jobs from one organization to another, that minimizes the total energy consumption.

Each organization has what is called a *selfish restriction*, that being a energy restriction that makes unfeasible any schedule that increase the energy consumption of that organization compared to what would be if the same organization was alone (even if the global energy consumption decrease with that schedule.)

An interesting aspect of this problem is that the energy consumption is given by a convex function on the speed of the processor, making its analysis significantly different from the original MOSP problem.

On Section 2 we formally define the problem. Section 3 shows that the general problem is NP-hard and that the ratio between the energetic consumption of solutions that respect the selfish constraint to the cost of solutions that does not respect may be unbounded for some instances of the problem. Heuristics for the problem with several organizations executing jobs that must meet a deadline are presented in Section 4, and their energy savings are experimentally analyzed in Section 5. Finally, some conclusion remarks are presented in Section 6.

2 Problem Description and Notations

The general problem studied in this paper is how to perform energy-aware scheduling on cooperative platforms formed by a federation of organizations. Different independent organizations, interconnected in a grid-like fashion, share resources and exchange jobs, expecting an improvement on their performance and costs. We are interested in studying how to redistribute the load between the organizations, decreasing the total energy-cost of the entire platform.

We call this problem the Energy-Aware Multi-Organization Scheduling Problem (MOSP-ENERGY), after the Multi-Organization Scheduling Problem (MOSP), that first studied scheduling on grid computing platforms. Formally, we define our cooperative platform as a federation of N organizations. Each organization $O^{(k)}$, $1 \leq k \leq N$, shares a machine that supports continuous *dynamic speed scaling* (*i.e.*, processors can operate at any arbitrary speed s that can be changed by the scheduler over time) and intend to execute $n^{(k)}$ jobs. A job $J_i^{(k)}$, $1 \leq i \leq n^{(k)}$, is defined by its release date $r_i^{(k)}$, its deadline $d_i^{(k)}$ and its processing volume $w_i^{(k)}$. The job with the biggest deadline of $O^{(k)}$ is defined as $d_{\max}^{(k)} = \max_i d_i^{(k)}$. Job preemption is allowed.

At a given time, if the chose speed is s, the power required to operate the processor is given by $P(s) = s^\alpha$, where α is a constant real number that depends on the type and model of the processor, usually with a value between 2 and 3. The energy consumption on one machine is given by the integral of $P(s)$ over time. The total energy consumption of the system is the sum of the power consumption of the machines of all organizations.

In order to encourage the creation of these cooperative platforms, we impose a hard constraint on the feasibility of the schedules: no organization can have its costs increased by cooperating. We call this the *selfish restriction* of the organizations. In other words, if an organization $O^{(k)}$ can execute its jobs consuming a total energy of $E_{\text{local}}^{(k)}$ only using its own machines, then a feasible schedule $\mathcal{S}$ must ensure that $E_{\mathcal{S}}^{(k)} \leq E_{\text{local}}^{(k)}$ (otherwise the organization could just leave the platform). The optimization problem to be solved can be stated as:

$$\text{minimize } E_{\mathcal{S}} \text{ subject to } E_{\mathcal{S}}^{(k)} \leq E_{\text{local}}^{(k)}, \forall k$$

3 Complexity Analysis

3.1 The Cost of Having Selfish Organizations

Respecting MOSP-ENERGY selfish restriction restrains the set of feasible schedules. This limitation have an impact on the quality of the optimal solutions. For the general (*i.e.*, without the selfish restriction) energy minimization problem for multiple machines, it is known that:

Lemma 1 (Albers et al. [2]). *For any set of jobs, the energy of an optimal schedule on m processors is at least $1/m^{\alpha-1}$ times that of an optimal schedule on one processor.*

The worst case for MOSP-ENERGY is when all but one organizations are idle. The overloaded organization may not be able to migrate its jobs to the others in order to respect the selfish restriction. The optimal solution without the selfish restriction would be able to redistribute the load among all the m machines. So, the following corollary holds:

Corollary 1. *The ratio between the best solution that respects* MOSP-ENERGY *selfishness restriction to the best solution that does not respect it is* $m^{\alpha-1}$.

3.2 Computational Complexity

This section studies how hard is to find an optimal solution for the MOSP-ENERGY problem. We study, without loss of generality, the simpler case with 1 machine per organization. Lets consider the decision version of the MOSP-ENERGY defined as follows:

Instance: a set of N organizations (for $1 \leq k \leq N$, organization $O^{(k)}$ has $n^{(k)}$ jobs and 1 processor with variable speed) and an integer K.

Question: does there exist a schedule $\mathcal{S}$ such that the selfish restriction $E_{\mathcal{S}}^{(k)} \leq E_{\text{local}}^{(k)}$ is respected for all $O^{(k)}$ and such that its total energy consumption $E_{\mathcal{S}}$ is less than or equal to K?

We will show that:

Theorem 1. *The* MOSP-ENERGY *problem is NP-Complete.*

Proof. It is straightforward to see that MOSP-ENERGY $\in NP$. Our proof is based on a reduction from the well-known PARTITION problem [6]:

Instance: a finite set of positive integers $A = \{a_1, \ldots, a_n\}$.

Question: is there two disjoint subsets A_1 and A_2 of A such that $\sum_{a_i \in A_1} a_i = \sum_{a_j \in A_2} a_j$?

Given an instance of PARTITION, we construct an instance of MOSP-ENERGY with $N = 2$ organizations as follows. Let t and t' be two integers representing two different deadlines where $t < t'$. Let D be an integer representing a processing volume; we will discuss their values later.

Organization $O^{(1)}$ has only one job, $J_1^{(1)}$, with $r_1^{(1)} = 0$, $d_1^{(1)} = t$ and $w_1^{(1)} = D$. Organization $O^{(2)}$ has $n+1$ jobs: $J_1^{(2)}, \ldots, J_{n+1}^{(2)}$. The first job of $O^{(2)}$ is identical to the one from $O^{(1)}$: $r_1^{(2)} = 0$, $d_1^{(2)} = t$, $w_1^{(2)} = D$. The remaining n jobs have $r_i^{(2)} = 0$, $d_i^{(2)} = t'$ and $w_i^{(2)} = a_i$.

Let $\beta = \sum_{a_i \in A} a_i$. We define an integer K as:

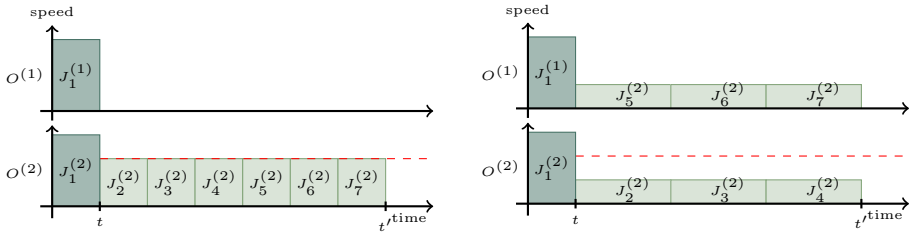
$$K = \frac{2D^{\alpha}}{t^{\alpha-1}} + \frac{\left(\frac{\beta}{2}\right)^{\alpha}}{(t'-t)^{\alpha-1}} + \frac{\left(\frac{\beta}{2}\right)^{\alpha}}{(t'-t)^{\alpha-1}}$$

And choose the values of D, t and t', such that: $\frac{D}{t} > \frac{D+\beta}{t'}$. Choosing $D \geq 5\beta$ and $t' \geq 3t + 1$ satisfy these conditions.

Now we can easily build an instance for MOSP-ENERGY from the set A in polynomial time, as depicted in Fig. 1(a). In this instance, the optimal local energy consumption of $O^{(1)}$ (computed by the YDS algorithm) is given by $E^{(1)}_{\text{local}} = \frac{D^\alpha}{t^{\alpha-1}}$.

Now, we will compute the cost of the local energy consumption of organization $O^{(2)}$. This cost can also be computed using the YDS algorithm. Recall from Section 1.1 that the optimal speed to execute a job is calculated using the concept of interval of maximum density, *i.e.*, the time interval such that the sum of the processing volumes of the jobs that start and finish in it, divided by the length of the interval, is maximum. This density is the speed on which the jobs inside this interval will be executed in the optimal schedule, hence, the total energy spent by a job is determined by its speed in an optimal schedule.

D, t and t' was chosen in such a way that $\frac{D}{t} > \frac{D+\beta}{t'}$. Thus, the interval of maximum density for both organizations will always be the interval on which the jobs of processing volume D and deadline t are. This means that in the optimal local schedule for $O^{(2)}$, job $J^{(2)}_1$ must be executed alone from time 0 until time t. From time t until time t', all the remaining jobs are executed. The energy spent by $O^{(2)}$ is then given by $E^{(2)}_{\text{local}} = \frac{D^\alpha}{t^{\alpha-1}} + \frac{\beta^\alpha}{(t'-t)^{\alpha-1}}$.



(a) Initial instance.

(b) Scheduling after the migration of some jobs from $O^{(2)}$ to $O^{(1)}$.

Fig. 1. Reduction of the MOSP-ENERGY problem from PARTITION

Now we must show that this transformation is a reduction. First, suppose that the set A can be split into two disjoint subsets A_1 and A_2 such that $\sum_{a_i \in A_1} a_i = \sum_{a_j \in A_2} a_j$. In order to respect the selfish restriction and avoid an increase on the local cost of organization $O^{(1)}$, neither the first job from $O^{(1)}$ nor $O^{(2)}$ can migrate. The only way to decrease the total energy cost is to migrate some of the other jobs. We will split the last n jobs of $O^{(2)}$ into 2 subsets, $\mathcal{J}_1$ and $\mathcal{J}_2$ such that if $a_i \in A_1$ than the job $J^{(2)}_{i+1}$, with $w^{(2)}_{i+1} = a_i$, belongs to set $\mathcal{J}_1$. Otherwise, it belongs to $\mathcal{J}_2$.

We can migrate the jobs of one of the subsets, say $\mathcal{J}_1$, to organization $O^{(1)}$. As consequence of our assumptions on D, t and t', the migrations does not change $E^{(1)}_{\text{local}}$. After all migrations, the cost of organization $O^{(2)}$ will be given by:

$$E_{\mathcal{S}}^{(2)} = \frac{D^\alpha}{t^{\alpha-1}} + \frac{\left(\sum_{J_i^{(2)} \in \mathcal{J}_1} w_i^{(2)}\right)^\alpha}{(t'-t)^{\alpha-1}} + \frac{\left(\sum_{J_j^{(2)} \in \mathcal{J}_2} w_j^{(2)}\right)^\alpha}{(t'-t)^{\alpha-1}}$$

Since $\sum_{J_i^{(2)} \in \mathcal{J}_1} w_i^{(2)} = \sum_{J_j^{(2)} \in \mathcal{J}_2} w_j^{(2)} = \frac{\beta}{2}$, the global energy consumption on this schedule is equal to:

$$E_{\mathcal{S}} = E_{\mathcal{S}}^{(1)} + E_{\mathcal{S}}^{(2)} = \frac{2D^\alpha}{t^{\alpha-1}} + \frac{2\left(\frac{\beta}{2}\right)^\alpha}{(t'-t)^{\alpha-1}} = K$$

Thus, the local constraints are respected and the total energy spent is K.

Suppose now that there is a valid schedule for this instance such that its total cost is less than or equal to K. It implies that some jobs from organization $O^{(2)}$ must have migrated. We can split the jobs from $O^{(2)}$ into two subsets $\mathcal{J}_1$ and $\mathcal{J}_2$ such that $J_i^{(2)} \in \mathcal{J}_1$ if job $J_i^{(2)}$ was migrated to $O^{(1)}$, otherwise $J_i^{(2)} \in \mathcal{J}_2$. Now, we split the set A in two subsets A_1 and A_2 in such a way that $a_i \in A_i$ if and only if $J_{i+1}^{(2)} \in \mathcal{J}_1$; otherwise, it belongs to $\mathcal{J}_2$. The global energy consumption of this schedule is given by:

$$E_{\mathcal{S}} = \frac{2D^\alpha}{t^{\alpha-1}} + \frac{\left(\sum_{J_i^{(2)} \in \mathcal{J}_1} w_i^{(2)}\right)^\alpha}{(t'-t)^{\alpha-1}} + \frac{\left(\sum_{J_j^{(2)} \in \mathcal{J}_2} w_j^{(2)}\right)^\alpha}{(t'-t)^{\alpha-1}}$$

Since $E_{\mathcal{S}} \leq K$, we deduce from the two previous equations that:

$$\begin{aligned} & \frac{\left(\sum_{J_i^{(2)} \in \mathcal{J}_1} w_i^{(2)}\right)^\alpha}{(t'-t)^{\alpha-1}} + \frac{\left(\sum_{J_j \in \mathcal{J}_2} w_j^{(2)}\right)^\alpha}{(t'-t)^{\alpha-1}} \leq \frac{2\left(\frac{\beta}{2}\right)^\alpha}{(t'-t)^{\alpha-1}} \\ \implies & \left(\sum_{J_i^{(2)} \in \mathcal{J}_1} w_i^{(2)}\right)^\alpha + \left(\sum_{J_j^{(2)} \in \mathcal{J}_2} w_j^{(2)}\right)^\alpha \leq 2\left(\frac{\beta}{2}\right)^\alpha \end{aligned}$$

Since $x^\alpha + y^\alpha$ is convex and $x + y = \beta$, then, by definition of convexity, the function $x^\alpha + y^\alpha$ is minimum when $x = y$ and $x^\alpha + y^\alpha \geq 2(\frac{\beta}{2})^\alpha$. In our case, this means that:

$$2\left(\frac{\beta}{2}\right)^\alpha \leq \left(\sum_{J_i^{(2)} \in \mathcal{J}_1} w_i^{(2)}\right)^\alpha + \left(\sum_{J_j^{(2)} \in \mathcal{J}_2} w_j^{(2)}\right)^\alpha \leq 2\left(\frac{\beta}{2}\right)^\alpha \tag{1}$$

Now, we split set A into two subsets A_1 and A_2 such that $a_i \in A_1$ if $J_i^{(2)} \in \mathcal{J}_1$; otherwise $a_i \in A_2$. From Eq. 1, $\sum_{a_i \in A_1} a_i = \sum_{J_i^{(2)} \in \mathcal{J}_1} w_i^{(2)} = \frac{\beta}{2}$.

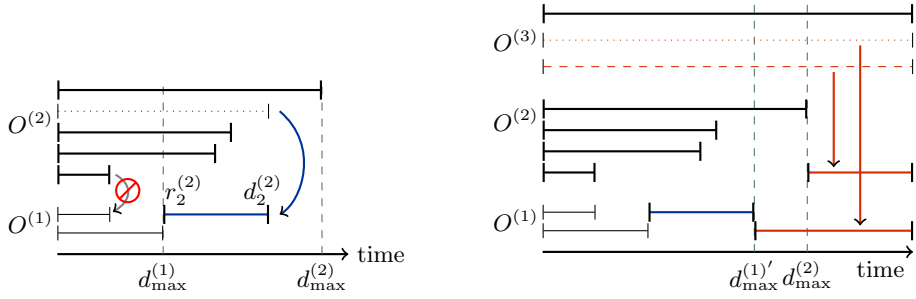
In other words, it means that $\sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i$. This proves that set A can be split into two disjoint subsets A_1 and A_2 such that $\sum_{a_i \in A_1} a_i = \sum_{a_j \in A_2} a_j$ if and only if there is a valid schedule to this instance such that its total cost is less than K. This concludes our proof.

4 Heuristics

We developed heuristics for the MOSP-ENERGY problem for instances of bag-of-tasks jobs that are available at the beginning of the batch ($r_i = 0$). Without loss of generality, we assume that all $w_i = 1$ and only deadlines are free to vary.

The main idea of these heuristics is to migrate jobs from a more costly organization to a less costly one, always respecting the selfish restrictions. This is achieved by adjusting the release date of the migrated jobs to values higher then the higher deadline of the host organization. If one migrates a job to an interval that overlaps with any job from the hosting organization, the processor may have to increase its speed to be able to respect all the deadlines, resulting in an increase of the energy cost to execute the jobs of hosting organization. This may happen if value of the maximum density interval is changed. Avoiding these migrations ensures that the energy to run the host's jobs will remain unchanged. Fig. 2(a) illustrates the idea, showing the result of a possible migration.

We start considering how to redistribute energy as a resource among $N = 2$ organizations and then present a generic heuristic for N organizations.



(a) $J_2^{(2)}$ is migrated to $O^{(1)}$ and has its release date adjusted to $d_{\max}^{(1)}$.

(b) $d_{\max}^{(1)}$ is adjusted and $O^{(3)}$ migrates jobs to $O^{(2)}$ and $O^{(1)}$.

Fig. 2. Schema of the heuristics migrations

4.1 Heuristics for $N = 2$ Organizations

Consider an instance of the MOSP-ENERGY problem with only $N = 2$ organizations. Assume, without loss of generality, that $d_{\max}^{(1)} \leq d_{\max}^{(2)}$.

Our heuristics — based on the YDS algorithm (see Section 1.1) — iteratively find the maximum density interval of the more costly organization on each iteration. After performing the migrations, we use the original YDS algorithm on each organization to compute the minimum processor speed to execute each job.

At each iteration, the heuristics compute the maximum density interval $[r_\Delta^{(2)}, d_\Delta^{(2)}]$ of the organization with the biggest $d_{\max}$ (in our case, $O^{(2)}$) and the list of jobs $J_{\Delta_i}^{(2)} \in \mathcal{J}_\Delta^{(2)}$ that lies inside it. We have three cases to consider:

(i) if $\mathbf{d}^{(\mathbf{2})}_{\mathbf{\Delta_i}} \leq \mathbf{d}^{(\mathbf{1})}_{\max}$ the heuristic cannot migrate $J^{(2)}_{\Delta_i}$ without increasing the energy spent by the other organization's local jobs;
(ii) if $\mathbf{r}^{(\mathbf{2})}_{\mathbf{\Delta_i}} \geq \mathbf{d}^{(\mathbf{1})}_{\max}$ the heuristic can migrate the job "as is" (without changing its release date and deadline). For $N = 2$, this case is equivalent to the problem for m machines and can be optimally solved on polynomial-time [2];
(iii) if $\mathbf{r}^{(\mathbf{2})}_{\mathbf{\Delta_i}} < \mathbf{d}^{(\mathbf{1})}_{\max}$ and $\mathbf{d}^{(\mathbf{2})}_{\mathbf{\Delta_i}} > \mathbf{d}^{(\mathbf{1})}_{\max}$ the job can be migrated, but its release date must be adjusted, has shown in Fig. 2.

Our heuristics differ on how to handle the third case, which we call the *border jobs*, since they intersects the border defined by $d^{(1)}_{\max}$. We will describe how each heuristic tackles the border problem in the following sections.

Greedy Heuristic. The first heuristic deals with the border jobs in a greedy way. At each iteration, we compute the maximum density interval of $O^{(2)}$. If the jobs on $\mathcal{J}^{(2)}_{\Delta}$ does not intersects the border, we solve the problem as explained before. If the jobs are in the border, we choose the job with biggest deadline. If the migration of this job (adjusting its release date to $d^{(1)}_{\max}$) decreases the total energy cost of the platform, the job is migrated. Otherwise, the job remains in its original state on $O^{(2)}$. We repeat this process until there are no more jobs to consider on $O^{(2)}$.

Probabilistic Heuristic. In this heuristic, the border is handled in a probabilistic way. A job $J^{(2)}_i \in \mathcal{J}^{(2)}_{\Delta}$ in the border is migrated with probability $p_i = \begin{cases} \frac{d^{(2)}_i - d^{(1)}_{\max}}{d^{(2)}_{\Delta} - d^{(1)}_{\max}} & \text{if } d^{(2)}_i > d^{(1)}_{\max} \\ 0 & \text{otherwise.} \end{cases}$

This heuristic has the advantage of being very fast in practice, whereas Greedy must run the YDS algorithm several times.

Brute-Force Heuristic. The border problem that we are trying to solve is, essentially, a problem of splitting the set $\mathcal{J}^{(2)}_{\Delta}$ into two disjoint subsets, migrating one to $O^{(1)}$. For small inputs, it is computationally feasible to try all possible splits. The results from the experiments with this approach gives insight into the quality of the solutions provided by the other heuristics.

Consider the subset of $\mathcal{J}^{(2)}_{\Delta}$ that is in the border. We enumerate all possible partitions of $\mathcal{J}^{(2)}_{\Delta}$ in two disjoint subsets (one set will be migrated and the other will remain on $O^{(2)}$) and test which one minimizes the total energy cost. This heuristic is, of course, exponential in the number of jobs in $\mathcal{J}^{(2)}_{\Delta}$.

4.2 Heuristic for N Organizations

Using the ideas presented on Section 4.1, we have designed a simple polynomial-time heuristic for the case when we have more than two organizations. The heuristic is based on the Iterative Load Balancing Algorithm (ILBA [5]).

The basic principle of our heuristic is to redistribute the energy expenditure of the organizations starting with the two organizations that have the smallest deadlines and iteratively add the jobs from the most costly organizations. One-by-one, each organization has its energy decreased.

The heuristic enumerates the organizations by non-decreasing values of their $d_{\max}$, *i.e.*, $d_{\max}^{(1)} \leq d_{\max}^{(2)} \leq \cdots \leq d_{\max}^{(N)}$ and considers, one-by-one, each organization $O^{(k)}$ for $k = \{2, \ldots, N\}$. The choice of which jobs from $O^{(k)}$ should be migrated is done based on the concept of the maximum density interval (MDI). The algorithm computes the MDI of its jobs and migrates the border job with biggest deadline to the organization among $O^{(1)}, \ldots, O^{(k-1)}$ that decreases the most the total energy.

When there is no more job worth migration on the density interval, the value of $d_{\max}$ of all organizations $O^{(1)}, \ldots, O^{(k-1)}$ is updated — see Fig. 2(b) — and the algorithm checks if there is a new MDI on $O^{(k)}$ with jobs worth migration. If yes, it repeats the migration process. If not, the algorithm will try to redistribute the jobs of the next organization ($O^{(k+1)}$).

This process is repeated until all organizations had been considered. Note that by updating the $d_{\max}$ value after considering each MDI, we never increase the energy spent to execute the jobs already scheduled. Consequently, MOSP-ENERGY selfish restrictions are always respected.

5 Experimental Evaluation

We designed a series of experiments to evaluate the heuristics presented on the previous section. The experiments were evaluated using randomly generated workloads akin to typical environment found on academic cooperative platforms [5]. We evaluated the algorithms with instances containing a random number of machines, organizations and jobs with different deadlines. Two different scenarios were considered.

In the first, the number of initial jobs in each organization follows a Zipf distribution with exponent equal to 1.4267 and the jobs' deadlines are uniformly distributed. In the second, the $C_{\max}$ of these organizations follows the same Zipf distribution, and $d_i^{(k)} = C_{\max}^{(k)}, \forall i, k$ and the jobs are uniformly distributed among the organizations. The intuition about the scenarios is that the first configuration best models the distribution of jobs among organizations in shared platforms [7], where the second models the selfish restriction of the original MOSP problem, with the deadlines representing the initial makespan of the organizations.

Table 1. Results for $N = 2$ organizations. For different numbers of jobs per organization, we show how each heuristic performs if compared to no cooperation at all.

# Jobs/Org	% Greedy	% Probabilistic	% Brute-Force
5	0.69	1.85	2.45
10	0.94	2.12	3.09
15	2.29	1.61	3.21
20	1.79	1.27	4.97
50	0.78	0.67	7.44
100	0.32	0.30	3.08

Table 2. Results for $N = 10$ and 20 organizations, showing how the iterative algorithm performs if compared to no cooperation at all

N	# Jobs/Org	Energy Saved (%)
10	5	11.87
10	10	6.81
10	15	5.47
10	20	4.64
10	30	2.86

N	# Jobs/Org	Energy Saved (%)
20	5	15.64
20	10	9.81
20	15	6.11
20	20	5.04
20	30	3.24

Table 3. Performance results for $N = 2$ organizations on the second scenario

# Jobs/Org	% Greedy	% Probabilistic	% Brute-Force
5	4.22	5.86	6.72
10	4.12	3.19	5.94
15	2.08	2.96	6.81

Tables 1 and 2 summarizes the results obtained by our heuristics for the first scenario. Our preliminary tests showed that the maximum $d_{\max}$ does not affect significantly the results for the first scenario. So, due to the lack of space, all results for this scenario are presented for $d_{\max} = 50$. Varying the number of jobs per organization, we show how much each heuristic can save on the total energy usage if compared to the total energy usage that could have been obtained without migrations (applying the YDS algorithm for each organization individually.) Each result is presented as the average of 200 experiments.

The results shows that for $N = 2$ organizations, the energy saving is limited by the selfish restriction of the organizations. The Greedy heuristic is able to save more energy than Probabilistic when the ratio between the number of jobs to the number of organizations is higher. The results obtained with Brute-Force are presented for the sake of comparison. For $N = 10$ and $N = 20$, our iterative algorithm was able to obtain savings up to 11.87% and 15.64%, respectively. Further investigation is needed for instances with higher number of jobs per organizations. In this case, the organizations have a higher probability of have similar $d_{\max}$. This fact hampers the ability of improving the solutions because of MOSP-ENERGY selfish restriction.

Tables 3 and 4 summarizes the results obtained by our heuristics for the second scenario. The results show a significant energy reduction — up to 27.45% — if the notion of deadline is related only to the initial makespan.

Table 4. Performance results for $N = 10$ and 20 organizations on the second scenario

N	# Jobs/Org	Energy Saved (%)
10	5	17.99
10	10	19.10
10	15	19.13

N	# Jobs/Org	Energy Saved (%)
20	5	20.08
20	10	25.50
20	15	27.45

6 Concluding Remarks

In this work, we have studied the problem of scheduling on cooperative platforms considering energy as a communal resource. The objective of the Energy-Aware Multi-Organization Scheduling Problem (MOSP-ENERGY) is to minimize the total energy consumption of the entire platform, while assuring that the energy cost to execute jobs from a particular organization will not increase.

Balancing energy consumption is significantly different from the load balancing problem because of the convexity of the cost function. We have proved that the MOSP-ENERGY problem is NP-hard and that the ratio between the best solution respecting the organizations' selfish restriction to the solution that minimized the total energy is equal to $m^{\alpha-1}$.

We have designed heuristics to show how one can redistribute the energy between organizations respecting the selfish restriction. Our experimentals shows that we can save as much as 27% energy of the total spent by the platform.

This study was a first step on a better understanding of the role of energy costs on cooperative platforms. Further research will investigate approximation algorithms for the problem and fairness issues on the distribution of the energy costs between the organizations even if the jobs from different organizations belong to the same maximum density interval.

References

1. Albers, S., Antoniadis, A., Greiner, G.: On multi-processor speed scaling with migration. In: ACM Symposium on Parallelism in Algorithms and Architectures, pp. 279–288 (2011)
2. Albers, S., Müller, F., Schmelzer, S.: Speed scaling on parallel processors. In: ACM Symposium on Parallel Algorithms and Architectures, pp. 289–298 (2007)
3. Cohen, J., Cordeiro, D., Trystram, D., Wagner, F.: Coordination mechanisms for selfish multi-organization scheduling. In: IEEE International Conference on High Performance Computing, pp. 1–9 (December 2011)
4. Cohen, J., Cordeiro, D., Trystram, D., Wagner, F.: Analysis of multi-organization scheduling algorithms. In: D'Ambra, P., Guarracino, M., Talia, D. (eds.) Euro-Par 2010, Part II. LNCS, vol. 6272, pp. 367–379. Springer, Heidelberg (2010)
5. Dutot, P.F., Pascual, F., Rzadca, K., Trystram, D.: Approximation algorithms for the multiorganization scheduling problem. IEEE Transactions on Parallel and Distributed Systems 22(11), 1888–1895 (2011)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman (January 1979)
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. 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 (2007)
9. Yao, F., Demers, A., Shenker, S.: A scheduling model for reduced CPU energy. In: Symposium on Foundations of Computer Science, pp. 374–382. IEEE (1995)