Green IT scheduling for data center powered with renewable energy

Léo Grange, Georges da Costa, Patricia Stolf

Green IT scheduling for data center powered with renewable energy

Léo Grange *, Georges Da Costa, Patricia Stolf

IRIT, University of Toulouse, 118 Route de Narbonne, F-31062 Toulouse Cedex 9, France

- A new scheduling algorithm aware of energy availability is proposed.
- Several heuristics are implemented and compared.
- Considering the electrical infrastructure as a black-box still lead to good results.
- The amount of freedom allowed by the SLA greatly affects the achievable saving.
- Simulations show a reduction of brown energy consumption up to 49%.

Keywords:
Renewable energy
Online scheduling
Data center
Energy aware scheduling

A B S T R A C T

In recent years, the question of the energy consumption of data centers has become more and more impor-tant, and several studies raised the possibility of using renewable energy to power them. The intermittent nature of commonly used renewable energy sources is a major drawback of using them directly on-site. In this paper, we present an approach for scheduling batch jobs with due date constraints, which takes into account the availability of the renewable energy to reduce the need of brown energy and therefore running cost. The approach we propose differs from the existing methods by providing a scheduling algorithm agnostic of the electrical infrastructure. A separated system, managing the renewable sources, provides an arbitrary objective function, which is used to guide the scheduling heuristic. We implemented our approach in a data center simulator, and evaluated it by considering a small-scale center powered with solar panels and connected to the electrical grid. The relationship between the flexibility allowed by the user negotiated SLAs and the behavior of the algorithm is studied, and compared to existing approaches from the literature. Our experiments show a reduction of brown energy consumption up to 49% and a cost saving up to 51%, compared to a traditional scheduler unaware of renewable availability.

1. Introduction

Today more than ever, the energy consumption generated by the growing use of information and communication technologies (ICT) is a major issue, from both economical and ecological points of view. The emergence and development of grid computing and cloud computing paradigms, during the last decade, caused the increase of data centers, as much in number than in size. Data centers are becoming a significant part of the global electrical consumption. Their total consumption in 2012 was estimated to almost 270 TWh [1]. This is roughly equivalent to 1.4% of the worldwide electrical consumption, while the complete ICT sector (excluding manufacturing) accounts for 4.7% of it. According to the same study, the data center power needs increased annually by 5% between 2006 and 2012. The projections for the next decade suggest higher growth rate in a near future. For the year 2030, the data centers alone may use between 3% (best case) and 13% (worst scenario) of the global electricity production [2].

Because of the current energy consumption of the ICT sector and its growing requirement, the responsibility of big companies in term of greenhouse gas emission and pollution in general is often pointed out. However, ICT firms seem involved in reducing their ecological impact more than other industry sectors by buying renewable energy for their needs, with several companies such as Intel or Adobe covering all their consumption in U.S. by this way [3]. This is also pointed out by Greenpeace, in their Click Clean report[1] claiming that, between 2011 and 2016, 16 of the major internet companies made "a meaningful long-term commitment to be 100% renewably powered". Their are many reasons for this recent change in companies behaviors. Radu [4] studied the determinants of Green ICT adoption in general, pointing out economic,

* Corresponding author.
E-mail addresses: leo.grange@irit.fr (L. Grange), dacosta@irit.fr (G. Da Costa), patricia.stolf@irit.fr (P. Stolf).

https://doi.org/10.1016/j.future.2018.03.049

[1] «Clicking Clean: Who is winning the race to build a green internet?», http://www.clickclean.org/downloads/ClickClean2016%20HiRes.pdf.

ethical and regulatory-related reasons, such as long term cost reduction, pro-environment grants, organization strategy and image.

The transition toward sustainable energy, not only for data centers, is one of the important global topic of this century. Krakowski et al. [5] study several scenarios for increased penetration of renewable energies in the French electrical mix. Multiple scenarios target 100% penetration in 2050, showing feasibility, cost and limitations of each. The authors also compare their results to a previous prospective study from the French environment and energy management agency (ADEME)[2] with similar objective but using different methods. Other agencies and researchers published similar studies for other regions, such as Elliston et al. for Australia [6]. These studies show that renewable energy at large scale is technically and economically realistic for the next decades. However, more work should be done in order to reach this goal as soon as possible, by studying not only how to produce this energy, but also how to build systems interacting well with such renewable sources.

Several operators have already built data centers at least partially powered with on-site renewable energies. For instance AISO.net, a small cloud service provider, uses on-site solar panels to run its data center. More recently, in 2018, Google announced the construction of a massive solar power plant producing 2.9 GWh annually on the Saint-Ghislain data center facility site, in Belgium. As most of the renewable energy sources, like solar and wind, are intermittent and difficult to predict in the long-term, new problematics related to energy management are raised. A lot of researches are currently done on this topic, and several perspectives are explored.

One of these perspectives is to take advantage of the geographical distribution of such data centers [7–9]. Indeed, the weather conditions between distant places are little correlated. By balancing the load across multiple data centers, it is possible to increase or decrease the power needed by each one depending on its local renewable production at a given time.

Another main perspective considers acting at a single data center level. It aims to manage the workload in such a way that power consumption matches as closely as possible to the power available through renewable energies. Depending on the considered workload, several methods may be considered. With interactive services, like web servers, it is possible to use traditional energy management techniques, which lead to act on their energy versus performance trade-off [10,11]. By considering a workload composed of batch jobs, these possibilities remain, and other ones are applicable. In particular, batch jobs may be delayed to some extent, in order to run when more energy will be available [12–15].

The work presented in this paper is focused on the latter area. Our approach is targeting the management of a single data center, with a workload composed of batch jobs with due date constraints, using on-site renewable energy sources. Contrary to existing approaches, detailed in Section 2, we propose to separate the optimizations of electrical infrastructure from the optimizations of the computing resources. We designed an online greedy scheduling algorithm (called Attractiveness-Based Blind Scheduling Heuristic, or ABBSH), which exchanges information with the electrical management system in order to take availability of energy in consideration. Along with this algorithm, several multi-objective functions are proposed to handle the trade-off between energy and performance considerations.

The approach has been implemented in a data center simulator from the community, which we extended to simulate various electrical components. To evaluate the proposed algorithm with realistic use cases, we have used a workload generator configured to mimic the statistical distributions of a real, large-scale cluster owned by Google. In addition, we have evaluated how the choice of the due date of the tasks impacts the performances of our algorithm. Finally, we compared the results of our approach to those of the GreenSlot scheduler [12], a renewable-aware approach from the literature.

The presented contributions are:

- a new scheduling heuristic for data center powered by renewable energy sources, with limited knowledge of the electrical sources infrastructure, detailed in Section 3.5
- an evaluation of the relationship between the constraints of QoS and the performances of scheduling algorithms for batch jobs, based on the results of our experiments in Section 6
- an algorithm for computing the lower bound for the energy consumed from the grid, presented in Section 4.5

This paper is organized as follows: in Section 2 an overview of the researches related to energy-aware and renewable-aware scheduling are presented. Section 3 contains a description of our approach, including an overview of the concepts used and a detailed description of the scheduling algorithm. The methodology used for validating our approach is described in Section 4. Section 5 contains the results of our experiments, which are interpreted and discussed in Section 6. Finally, we present our conclusion in Section 7.

2. Related work

To understand better the complexity and the diversity of approaches for reducing the ecological impact of data centers, it is required to look at several research areas. This section reviews related works, from the traditional management of data centers to the different explored perspectives for exploiting renewable energy sources inside them.

Data center management and quality metrics. Even without considering energy consumption, the field of data center management is huge and very active. Researches focus on different aspects of data centers, either to improve the quality of service of consumers' jobs or to improve the resources usage. Jennings and Stadler [16] give a good overview of the existing works targeting Cloud data centers and the main challenges in the management of such infrastructures. The authors highlighted 8 functional areas in this field, including two related to scheduling of resources and one related to resource pricing and profit maximization. The approach presented here, while not specifically designed for virtualized environments, belongs to those research areas.

A precise evaluation of the quality of a decision in a data center is complex. One of the main problems is to have a metric independent from the workload to compare different decisions. Even in specific fields, obtaining this metric is challenging. In [17] authors propose a complete evaluation of value creation in the context of big data companies. They conclude on the fact that the current works are preliminary and insufficient to create a dedicated metric to the specific field of big data. Guérout et al. [18] define multiple QoS metrics for Software as a Service Cloud platforms, which they used in [19] as a base for objective formulation of a scheduling approach. They are organized in four categories: performance, dependability (such as reliability or availability), security and cost (including both the cost for the consumer, the energy cost for the provider and the carbon emission cost). In the following of this article, we consider only three metrics:

- the respect of the *due date* associated to a task, which is a performance QoS metric and a part of the Service Level Agreement;

[2] «A 100% renewable electricity mix? Analyses and optimisations», http://www.ademe.fr/en/a-100-renewable-electricity-mix-analyses-and-optimisations.

- the amount of *energy consumed* from the electrical grid (not provided by the on-site renewable energy sources);
- the *cost of this energy*, which depends on the electrical provider pricing with changes over the day.

By studying different distributions of due dates in the tasks submitted to the data center, we also study how this SLA constraint impacts the other quality metrics.

Energy-aware scheduling for data centers. As it impacts directly the exploitation cost, many current researches on data center management take energy consumption into account. Multiple ways of saving energy are currently explored [20], from hardware improvement to networking management optimizations and more efficient cooling systems. For the specific field of data center scheduling, there are numerous energy-efficient approaches already proposed, each leveraging one or more techniques to save energy, without dealing with renewable energy specifically.

One of the common way to reduce the energy consumption is to make use of Dynamic Voltage and Frequency Scaling (DVFS) to act on the performance/consumption trade-off. Wang et al. [21] propose several heuristics for scheduling parallel tasks with precedence constraints, using DVFS to reduce consumption generated by non-critical tasks, without increasing the total execution time of the set of tasks. In addition, they present a mechanism based on a System Level Agreement (SLA) established with the customer, allowing to reduce the energy needed even more in exchange for a bounded increase of execution time. Von Laszewski et al. [22] present a virtual machine scheduler for homogeneous, cloud-oriented clusters. They use DVFS to reduce the power consumption of the physical machines, depending on the QoS requirements of the virtual machines running on them. Contrary to those approaches, the work presented here does not use DVFS. Instead, to reduce the power consumption, we rely on shutting down the idle machines.

With the increased use of virtualization in data centers, some recent approaches leverage the advantages of such environment to reduce the energy consumption. Zhou et al. [23] present a greedy heuristic for minimizing the energy consumption using consolidation while considering the quality of service of the virtual machines (VMs). The QoS metrics consist in resource requirements of VMs and in performance degradation caused by VM migration during consolidation. By measuring the load of each machine, the algorithm try to keep it bounded to avoid both overloading to preserve VMs' resources and under-loading to reduce power consumption. While our approach also try to avoid under-loading of machines, we do not leverage virtualization techniques and therefore do not use live migration and consolidation for this purpose.

Energy- and thermal-aware scheduling algorithms are often studied in the context of data centers. Sun et al. [24] give a spatio-temporal thermal model for machines in data centers. They use this model to develop a thermal-aware online scheduler taking thermal constraints into account, using DVFS to dynamically change the power and thermal impact of the machines. In [25], authors propose several greedy algorithms to improve the cooling infrastructure efficiency and to take into account the high failure rate of large data centers. In our approach, we do not consider cooling systems. However those studies are complementary with our, as we focus more on the cooperation between energy sources and servers themselves.

Some works do not try to minimize the energy consumption but the *energy cost*. Li et al. [26] present a scheduling approach to minimize the cost of the energy consumed while guaranteeing security-related QoS. The security guarantee is obtained by adding security services at runtime according to the level of SLA of each task. The scheduling algorithm itself is an online heuristic, based on Lyapunov optimization. With only knowledge of the past and current electricity prices, the algorithm takes the decision of whether executing a task or waiting for cheaper electricity. The algorithm proposed in this paper is also an online scheduler. In addition, both approaches consider variation of grid electricity pricing, although we study simplest and more predictable patterns (on-peak/off-peak scheme). However, our approach takes advantage of renewable power availability forecasts in the near future to take a decision and is able to plan ahead the execution of submitted tasks, resulting in a more predictable resources usage.

Our approach differs from most of the energy-efficient schedulers as the first goal is not to reduce to total energy consumed, but the *non-renewable* energy consumed. Therefore, it has more similarities with approaches such as [26], which tries to minimize the energy cost while having dynamic electricity pricing.

Data center management with renewable energy. As stated previously, reducing the power consumption of a data center is different from improving the use of renewable energy sources. The latter is also actively studied from the last decade, therefore there have been a lot of research done on this topic.

The survey [27] presents and classify dozens of works related to this area. It also summarizes the different ways to address the problem and the two main open issues in the community, which are:

1. whether to use *on-site* or *off-site* energy sources
2. whether to exploit *temporal* or *spatial* diversity and flexibility in the data center workload.

In the scope of our work, we explore the on-site renewable energy sources case and try to take advantage of the temporal diversity and flexibility of the workload.

Several works focus on geographically distributed data center around the world [7–9], each having either on-site or off-site renewable energy sources. This perspective allows to exploit the spatial aspects of the workload. Gu et al. [28] propose an approach for serving Cloud user requests with such distributed data centers with renewable energy sources and energy storage devices. Several heuristics are presented to select a data center and a machine, with two different objectives: minimization of the total cost or minimization of the carbon emissions. In addition to the geo-distributed aspect, this work differs greatly from our by considering an *interactive* workload: a request should be served instantaneously and always takes the same short amount of time. At the contrary, we consider a workload of batch jobs, each taking a significant amount of time and deferrable until a specified due date.

Batch jobs scheduling with renewable energy. As the diversity of the approaches for leveraging renewable energies in data center is important, we propose to focus on some of them which consider a model quite close to the one we study.

Li et al. [29] propose PIKA, a scheduling infrastructure able to leverage renewable energy sources in data centers with a mix of interactive web services and batch jobs. Each job, both batch and interactive, are executed in a virtual machine (VM). While the jobs of the interactive part are executed immediately, the batch ones may be delayed in a queue until enough renewable energy is available or until it must be executed in order to respect its deadline. This approach uses VM migration to minimize the number of physical machines needed to execute the workload at a given time. The unused nodes are powered off to save as much energy as possible. In our approach, the workload is composed of batch jobs only and the jobs are executed directly on the physical machines. Therefore we do not use virtualization methods for consolidation purpose. Instead, our approach can plan the execution of jobs in the future, thus anticipating the use of each machine and reducing the number of powered-on machines.

With their approach named GreenSlot, Goiri et al. [12] considered a small cluster used for scientific computation, and powered partially with solar panels. Using prediction of renewable power available, along with grid electricity price, the GreenSlot algorithm delays jobs to execute them when the cost is the lowest (both in terms of brown energy usage and in terms of purchasing cost). The managed jobs have an estimated execution time and a deadline. This approach use a greedy heuristic, based on a discretization of future time into *slots* of fixed duration. Each slot is valuated with predicted renewable energy production, grid electricity cost, and number of available computing node at this time. To place a task, the method presented by Goiri et al. consists in testing each possible starting slot. The first one to allow using only renewable energy during each required slot is chosen, if any. Otherwise, the heuristic consists in choosing the one with minimal estimated electricity purchasing cost.

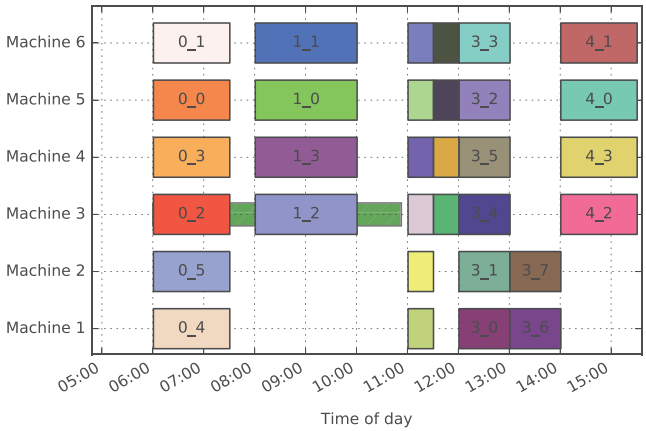
In another paper, Goiri et al. [13] present the GreenSwitch approach, along with a real data center prototype named Parasol in which they implemented and tested their algorithm. This data center, power by solar panels and batteries, is a small scale one, containing 64 energy-efficient machines. GreenSwitch is intended to manage MapReduce workloads, and specifically those using the Hadoop implementation. The authors used a Mixed Integer Linear Programming formulation, as a single optimization problem, of both workload scheduling and energy distribution constraints. Several variants of the optimization problem are studied, to explore the impact of using batteries or net-metering. Two different kinds of workload were used: deferrable, which can be assimilated to batch jobs that may be delayed until the end of the experiment window, and non-deferrable, which cannot be delayed. The results presented in [13] show that the GreenSwitch algorithm can significantly increase the use of renewable energy, particularly when the workload is deferrable, compared to a traditional Hadoop cluster.

Both of those approaches use a centralized optimization, which has a total knowledge of electrical and IT models, and takes care of both infrastructures. Our approach, instead, considers each of those two parts managed by its own optimization loop, and having only partial information on the other part. When studying deferrable workloads, GreenSwitch [13] considers a single deadline shared by all the tasks submitted during a day. At the contrary, the scheduler we propose manages individual due date for each task.

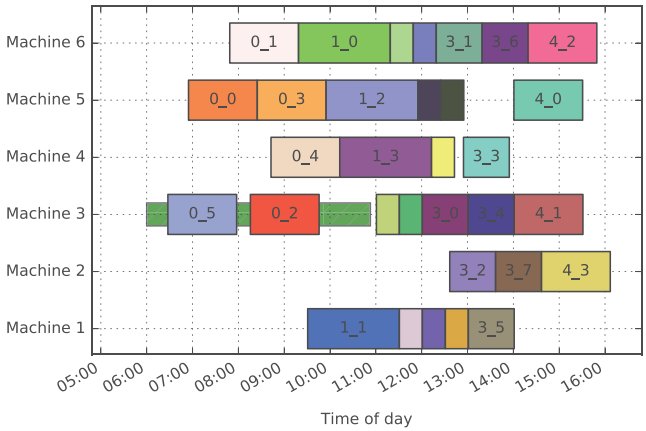
Summary of existing approaches. The diversity of the works presented in this section shows both the interest in the energy optimizations of data centers and the variety of approaches developed in this area. Table 1 summarizes the main characteristics of those approaches. It allows to highlight the similarities and differences between them and the approach proposed in the present paper, ABBSH, which is detailed deeply in the next section.

This diversity of approaches is also a challenge for evaluating new works. There are many, very different models of data centers, electrical infrastructures and workloads. While some of the differences do not avoid comparison of results between approaches, most make it either difficult or meaningless. For instance, PIKA [29] has many similarities with our work. However, by leveraging virtualization techniques and by using a mix of interactive web services and batch jobs, an experimental comparison with ABBSH would be either awkward or unfair.

The GreenSlot scheduling algorithm [12] has a model similar to our approach, as summarized in Table 1. It considers a single data center with on-site renewable energy sources and manages a workload of batch tasks with individual due dates. In addition, it has similar objectives (minimization of the non-renewable energy used and of the cost of this energy), making easier the comparison of the results. For these reasons, we compared experimentally our approach to GreenSlot, which appears to be the only existing approach usable in a such comparison.



(a) Task execution with traditional scheduler (first-fit).



(b) A possible execution with a scheduling approach aware of renewable energy availability.

Fig. 1. Task execution over time, for each machine, with and without awareness of renewable energy availability for the scheduling. The time between submission and due date is materialized, for the task '0_2', as an hatched rectangle.

One of the original features of ABBSH, its decision model, must be emphasized here. While the other works try to solve their optimization problem in a fully centralized way, we propose instead to have two parts communicating with each other. Each part (electrical and IT) does domain-specific optimizations without knowledge of the model and characteristics of the other part.

3. Proposed approach

In this section, we will present the concepts used by this approach, along with the system model considered. Then we will detail the scheduling algorithm itself and the several proposed heuristics.

3.1. Overview of renewable-aware batch scheduler

First of all, to give a better overview of our approach, we propose to illustrate how a renewable-aware scheduler differs from a traditional one. To show that, we consider a model similar to the one used in our approach, which will be detailed later. In this example, a data center is powered by a set of solar panels in addition to an electrical grid power supply. The workload consists in a set of batch tasks, each having a *due date* at which it should be completed. As the tasks are submitted over time by the users of the data center, they are not known by the system before submission.

Table 1
Summary of characteristics for existing energy-related data center (DC) scheduling works and comparison with the proposed approach. In the *Workload type* column, a distinction is made between batch jobs with *due dates* (the job should be executed before, but in some circumstances the job may finish after) and *deadline* (the job is considered as failed if the deadline is reached). For some approaches, the electrical infrastructure was not studied by the authors and therefore noted as *Grid (unspecified)*.

Approach	Energy-related objectives	Single or multiple DC	Electrical infrastructure	Workload type	Evaluation method	Method type	Decision model
Wang et al. [21]	Min. energy	Single	Grid (unspecified)	Batch with precedence constraints	Simulations	Greedy heuristic	Centralized
von Laszewski et al. [22]	Min. energy	Single	Grid (unspecified)	Batch (virtual machines) without due dates	Simulations	Greedy heuristic	Centralized
Zhou et al. [23]	Min. energy	Single	Grid (unspecified)	Interactive services (virtual machines)	Simulations (CloudSim)	Greedy heuristic	Centralized
Sun et al. [24]	Min. cooling energy	Single	Grid (unspecified)	Batch without due dates	Simulations	Greedy heuristic	Centralized
Li et al. [25]	Min. energy (including cooling)	Single	Grid (unspecified)	Batch with individual deadlines	Simulations (CloudSim)	Greedy heuristic	Centralized
Li et al. [26]	Min. energy cost	Single	Grid (unspecified)	Batch without due dates	Simulations	Heuristic (Lyapunov optimization)	Centralized
Gu et al. [28]	Min. energy cost or carbon emission	Multiple (geo-distributed)	Photovoltaic, wind turbines, grid (dynamic price)	Interactive services	Simulations	Greedy heuristic	Centralized
PIKA [29]	Min. non-renewable energy	Single	Photovoltaic, grid (without price information)	Mix of batch (virtual machines) with individual deadlines and interactive services	Simulations	Greedy heuristic	Centralized
GreenSlot [12]	Min. non-renewable energy or total energy cost	Single	Photovoltaic, grid (dynamic price)	Batch with individual due dates	Real testbed	Greedy heuristic	Centralized
GreenSwitch [13]	Min. non-renewable energy cost	Single	Photovoltaic, grid (dynamic price), batteries	Batch (MapReduce) jobs with single common deadline	Real testbed	MILP	Centralized
ABBSH (proposed)	Min. non-renewable energy or total energy cost	Single	Photovoltaic, grid (dynamic price)	Batch with individual due dates	Simulations (DCworms)	Greedy heuristic	Separated electrical and IT optimizations with cooperation

The scheduler should assign each task to a machine with enough resources to execute it, and has to respect as much as possible the due date required by the user.

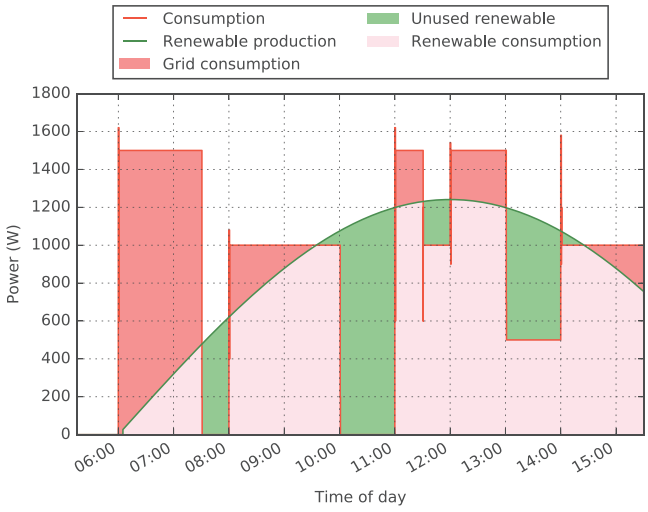
With a traditional energy-aware scheduler, several techniques exist to reduce the global energy consumption of the data center, such as DVFS management or powering off idle machines. Let us consider, first, a simple first-fit scheduler, able to power off idle machines. For a given workload, it will execute the submitted tasks as soon as possible, resulting in the execution trace presented in Fig. 1(a). This figure represents, for each machine, the tasks executed on it over time. Each task execution is represented as a rectangle, from its start to its completion. For one of those tasks, named '0_2', we represented additionally the time range between its submission and its due date as a thinner, hatched rectangle. In this example, the task '0_2' appears to end far before its due date, as the first fit scheduler executes it immediately after submission.

This scheduler is unaware of the origin of the energy used by the data center. Its power consumption for that scheduling, and the production of the solar panels over time, is shown in Fig. 2(a). Obviously, the consumption is not correlated with renewable energy availability. An important part of the energy, in red on this figure, is drawn from the grid. We also note that a part of the produced renewable energy, represented in green, is not consumed by the data center.

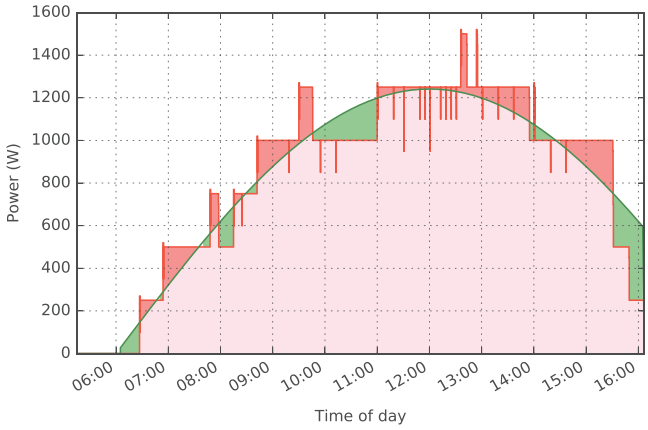
By considering a scheduler aware of the renewable energy availability, this information can be used to match more closely both consumption and production. Fig. 1(b) gives a possible schedule, for the same workload, which also respects all the due dates. However, almost all tasks are more or less delayed compared to the result of a first-fit algorithm. This is illustrated for the task '0_2', which is executed a couple of hours latter, but still ends before its due date. By scheduling tasks this way, the data center consumption over time is quite different from the previous schedule, as we can see in Fig. 2(b). It still does not fit perfectly the renewable energy production, but is clearly more correlated with the latter. Note that the total energy consumed by the data center is almost identical in both cases: what changes is how the power use is distributed over time and its source.

3.2. Data center infrastructure

One of the main goal of the presented approach is to keep separate, as much as possible, the electrical concerns from the computing ones. Therefore, we consider two distinct management systems. The first one is in charge of managing the electrical infrastructure (power sources, storage, and power distribution elements), in order to minimize the power losses and to maximize the usage of renewable energies when a given amount of power is to



(a) Power profile with traditional scheduler (first-fit).



(b) Power profile with a scheduling approach aware of renewable energy availability.

Fig. 2. Power profile (production of renewable sources and data center consumption) for a simple scenario, with a scheduler aware of renewable energy availability and with a traditional scheduler. Legend applies for both figures.

be supplied. The second one is the computing resources (shortened IT) management, which includes the data center scheduler and the management of the power states of individual machines. In order to reach a better global optimization of energy use, each management system needs to exchange pieces of information with the other. For instance, with some knowledge about the available renewable power during the near future, the computing management system may adjust the data center consumption in order to bring it closer to what will be available. To keep a low coupling between the two parts, the information exchanged is limited to a small subset. A cost indicator, named *attractiveness*, is used to abstract the optimization objectives of each system to the other, and will be presented in Section 3.4.

This design, compared to a single optimization loop in charge of both electrical and IT management, allows more flexibility in the considered infrastructures. Indeed, a change in the electrical components themselves or in the electrical optimization scheme would not impact the computing resources management, and vice versa.

An overview of the proposed infrastructure is shown in Fig. 3. The presented work is mainly focused on the IT optimization part, highlighted in green. Therefore, the electrical infrastructure and its

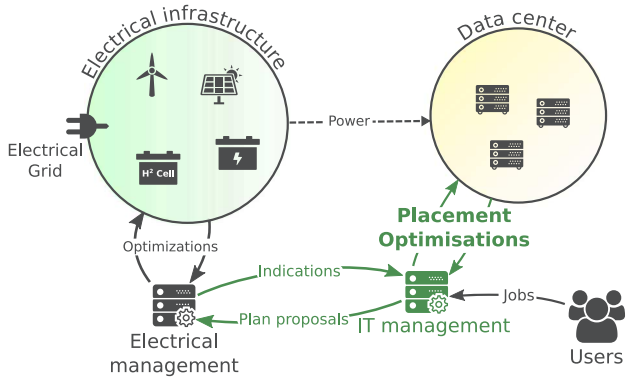


Fig. 3. Representation of the proposed infrastructure for both computing and electrical parts of a data center.

management system are highly simplified in this paper. However we show the flexibility of our approach regarding the electrical part by evaluating two different objective functions used in the electrical management system.

3.3. IT and electrical models

Across this section, we will detail formally the models used in the presented approach. As our work is focused on the IT part, the electrical model used here is intended to be simple and illustrative.

3.3.1. Data center model

We consider $\mathcal{J}$ the set of the J tasks submitted to the system at any moment. For each task j, we have the following information: its submission date S_j, its due date D_j, its normalized execution time T_j, the number of processors required $n_j \in \mathbb{N}^*$ and the amount of RAM needed r_j. In addition, the date at which the task starts its execution is noted B_j.

The computing resources are a set $\mathcal{M}$ of M machines, which may be heterogeneous. Each machine m is associated to its number of processors[3] N_m, the relative computation speed of the installed processors C_m, and its amount of usable memory R_m. The relative computation speed is used to compute the execution time of a task on this processor, such as the time needed for a task j is given by T_j/C_m. The static power consumption of a machine (excluding the processors one) is noted ps_m. The consumption of each processor is $pmin_m$ for the idle power and $pmax_m$ for its maximum (full load) consumption.

The use of a machine m over time is represented by several functions. The number of processors used at time t is given by $up_m(t)$ and the memory used by $ur_m(t)$. Each machine may be either powered on, off, or may be booting up or shutting down, which is represented by its state $s_m(t) \in \{On, Off, Boot, Shutdown\}$. For managing transitions between on and off states, additional information is needed for each machine. The booting up time $tboot_m$ and the averaged power consumption during boot $pboot_m$, with similar information for shutting down, respectively $tshut_m$ and $pshut_m$. It is then possible to get the instantaneous power

[3] We consider a processor as a physical execution core, assuming no logical core using SMT technologies. In addition, we make the assumption of an homogeneous set of processors for a given machine.

consumption of a machine at any time, $p_m(t)$, such as:

$$p_m(t) = \begin{cases} ps_m + up_m(t)pmax_m & \\ +(N_m - up_m(t))pmin_m, & \text{if } s_m(t) = On \\ pboot_m, & \text{if } s_m(t) = Boot \\ pshut_m, & \text{if } s_m(t) = Shutdown \\ 0, & \text{otherwise.} \end{cases}$$

We can therefore obtain trivially the total consumption of the machines of the data center with Eq. (1).

$$P(t) = \sum_m p_m(t). \tag{1}$$

A task is executed on a single machine (no migration), and without interruption (no pause and resume nor preemption). Given $\mathcal{E}$ the set of couples of task and machine on which it is executed, we have the following relationships.

$$\forall j, \exists m \quad (j, m) \in \mathcal{E}$$
$$\forall j \quad (j, m1) \in \mathcal{E} \wedge (j, m2) \in \mathcal{E} \iff m1 = m2.$$

To link the task execution model to the use of resources, we define some relationships.

$$\mathcal{R}_m(t) = \left\{ j | (j, m) \in \mathcal{E} \wedge t \geq B_j \wedge t < B_j + \frac{T_j}{C_m} \right\} \tag{2a}$$

$$up_m(t) = \sum_j^{\mathcal{R}_m(t)} n_j \tag{2b}$$

$$ur_m(t) = \sum_j^{\mathcal{R}_m(t)} m_j \tag{2c}$$

$$\forall t, \forall m \quad up_m(t) \leq N_m \tag{2d}$$

$$\forall t, \forall m \quad ur_m(t) \leq M_m \tag{2e}$$

$$\forall t, \forall m \quad s_m(t) \neq On \implies up_m(t) = 0 \wedge ur_m(t) = 0. \tag{2f}$$

Eq. (2a) defines, for a given time, the set of tasks executed on a given machine. Eqs. (2b) and (2c) represent the use of processors and memory caused by the execution of the tasks. Those resource usages are additionally constrained by the total available ones, as shown in Eqs. (2d) and (2e). Finally, Eq. (2f) describes that a machine which is powered off should have neither CPU nor memory load.

In this model, the due dates of the tasks are considered to be part of the SLA agreed by the customers. The respect of the promised quality of service for a task consists in finishing it before its due date, as formalized in Eq. (3a). As we use it as a metric, the average SLA violation for a given scheduling is given by Eq. (3b).

$$(j, m) \in \mathcal{E}, \quad q_j = \begin{cases} 1 & \text{if } S_j + \frac{T_j}{C_m} \leq D_j \\ 0 & \text{otherwise} \end{cases} \tag{3a}$$

$$SLA_{violation} = 1 - \frac{\sum_j q_j}{J}. \tag{3b}$$

3.3.2. Electrical model

The electrical infrastructure considered here is composed of an electrical grid power supply and a set of renewable power sources. Although we used solar panels in our experiments, no particular assumption on the kind of source is used in our model.

Let $\mathcal{W}$ be the set of the W renewable power sources. The effective available power at a time t for a source w is obtained through $available_w(t)$, and is supposed to be known only for the current instant. For each source, we have a prediction function $predicted_w(t)$, which gives an estimation of the available power for a time t in the future. The prediction is available only for t within a given time window $window_{prediction}$ from the current instant. The prediction function is considered as a black box, and numerous existing works may be used to compute it based on source characteristics and weather forecast, as much for solar panels [30,31] than for wind turbines [32,33].

The grid model allows to represent variations of the electricity price over time, such as on-peak/off-peak pricing. Pricing should be known in advance (within the same time window $window_{prediction}$), and is represented by a function $gridp(t)$, which is the price of a kilowatt-hour of grid energy at the time t.

3.4. Abstraction of electrical and IT objectives

Each of the management systems, electrical and IT ones, has only a full knowledge of the their own model, described above. Those two parts have therefore different objectives: the electrical part goal is to reduce either the total cost of purchased energy or the amount of grid energy used, and the IT goal is to schedule each submitted task with respect to the SLA.

To exchange bits of information between them, our approach consists in valuing these objectives in an abstracted and normalized way, named *attractiveness*. An attractiveness value is a real $\in [-1, 1]$ which represents, for a given *proposal*, its benefit or cost for one of the optimization loops. A value of -1 is used for the strongest cost, or an impossibility to accept a proposal, whereas a value of 1 corresponds to the most important benefit possible. The values between allow for any intermediate degree of cost or benefit, with 0 representing a neutral proposal.

We propose attractiveness functions for both electrical and computing parts, which associate a proposal to an attractiveness value, using information from its own model. As the attractiveness value abstracts the internal objectives of each part, it is possible to use it to guide an optimization, without the awareness of the elements considered to obtain it. Those values will be used by the scheduling algorithm to reach the best schedule.

3.4.1. Electrical attractiveness

The proposal sent by the IT optimization loop is to be seen as a part of a consumption plan. It contains an average power needed, the time at which the power is required, and the duration of the request. The electrical attractiveness is therefore calculated based on these proposal data and on the prediction of the model described in Section 3.3.2.

In order to demonstrate the flexibility offered by the use of attractiveness to abstract the internal objectives of each optimization loop, we propose two variants of the electrical attractiveness function. Whereas the first one (named variant A) only considers renewable energy use, the second one (variant B) also takes the grid electricity price into account.

The variant A is detailed in Algorithm 1. This function has several interesting properties. It is defined as a piecewise function, depending on the value of Δ_{power}. The two sub-functions, for $\Delta_{power} \geq 0$ and $\Delta_{power} < 0$ are identical at the exception of the constants used as *base*, *maxIncrement* and *rate*. In those functions, the value of Δ_{power} determines the impact of the *maxIncrement* factor. The closer Δ_{power} is to 0, the closer the resulting attractiveness is to *base*. At the opposite, when Δ_{power} approaches $\pm\infty$, the attractiveness asymptotically approaches $base + maxIncrement$. Both sub-functions are monotonic in their definition interval. Consequently, the computed attractiveness is bounded in $[\alpha_{pos}, \alpha_{pos} + \beta_{pos}]$ when $\Delta_{power} \geq 0$, and by $[\alpha_{neg}, \alpha_{neg} + \beta_{neg}]$ otherwise. Finally, the speed at which both sub-functions go from *base* to $base + maxIncrement$ is controlled by the value of *rate*.

Table 2
Values used for constants in Algorithms 1 and 2. Values of h_{pos}, h_{neg}, λ, θ_{min} and θ_{max} depend on the considered electrical infrastructure.

	α_{pos}	β_{pos}	α_{neg}	β_{neg}
Variant A	0.6	0.4	−0.7	−0.3
Variant B	0.6	0.4	N.A.	−0.3

Algorithm 2 is the attractiveness function for the variant B. It is similar to the variant A, but using a more complex function to determine the value of *base*, using the average price of the grid energy used (C_{grid}). It depends on three other constants, which are the minimum and maximum price of grid energy (respectively θ_{min} and θ_{max}), and λ. The value of λ, called *price factor* thereafter, determines the variation of *base* between $C_{grid} = \theta_{min}$ and $C_{grid} = \theta_{max}$. This modification impacts the value of the attractiveness when $\Delta_{power} < 0$ (when the average renewable energy production is lower than the requirement). Consequently, the more the grid energy price is expensive, the less the value of *base* is.

Some of the constants used in Algorithms 1 and 2 are given in Table 2. The other constants, which depend on the electrical infrastructure, will be given in Section 4.4.

Data:
$P_{available}$: predicted average available renewable power during the requested period
$P_{required}$: average required power during the same period
Result: a_{elec} : electrical attractiveness for this request $\in [-1, 1]$
$\Delta_{power} \leftarrow P_{available} - P_{required}$
if $\Delta_{power} \geq 0$ **then**
  $base \leftarrow \alpha_{pos}$;
  $maxIncrement \leftarrow \beta_{pos}$;
  $rate \leftarrow h_{pos}$;
else
  $base \leftarrow \alpha_{neg}$;
  $maxIncrement \leftarrow \beta_{neg}$;
  $rate \leftarrow -h_{neg}$;
end
$a_{elec} \leftarrow base + maxIncrement . \frac{\Delta_{power}}{\Delta_{power} + rate}$;
Algorithm 1: Electrical attractiveness function for a given consumption request (variant A).

3.4.2. IT attractiveness

For each proposed task schedule, the calculated IT attractiveness depends on the schedule start time related to the due date of the task ($t_{due} = D_j - T_j$), as shown in Algorithm 3. The purpose of this attractiveness function is to favor an early execution, and, more importantly, to penalize a schedule which leads to violate the due date. The difference between t_{due} and the submission date S_j is named *flexibility* of the task ($flexibility = D_j - T_j - S_j$). Indeed, it represents the amount of time during which the start of the task may be scheduled without violating the SLA. A task with low flexibility gives few freedom to the scheduler for choosing an appropriate time for its execution, compared to one with a higher flexibility. Two additional times are defined in addition to t_{due}, before and after it, named t_{urgent} and t_{late}. They are used respectively to reduce the attractiveness shortly before the deadline, and to encourage a schedule shortly after the due date over one occurring long time after. Table 3 presents the values used for the several constants required by Algorithm 3 to compute the IT attractiveness.

3.5. Scheduling algorithm

The purpose of a scheduling algorithm for batch tasks is to find, for each task j, a machine in $\mathcal{M}$ on which to run the task

Data:
$P_{available}$: predicted average available renewable power during the requested period
C_{grid} : average grid purchasing cost when renewable is not sufficient
$P_{required}$: average required power during the same period
Result: a_{elec} : electrical attractiveness for this request $\in [-1, 1]$
$\Delta_{power} \leftarrow P_{available} - P_{required}$
if $\Delta_{power} \geq 0$ **then**
  $base \leftarrow \alpha_{pos}$;
  $maxIncrement \leftarrow \beta_{pos}$;
  $rate \leftarrow h_{pos}$;
else
  $base \leftarrow -1.0 - \beta_{neg} - \lambda . \left(\frac{C_{grid} - \theta_{min}}{\theta_{max} - \theta_{min}} - 1 \right)$;
  $maxIncrement \leftarrow \beta_{neg}$;
  $rate \leftarrow -h_{neg}$;
end
$a_{elec} \leftarrow base + maxIncrement . \frac{\Delta_{power}}{\Delta_{power} + rate}$;
Algorithm 2: Electrical attractiveness function for a given consumption request, with grid price taken into account (variant B).

Data:
B_j : beginning of the proposed schedule
S_j : submission date
$t_{due} : D_j - T_j$
Result: a_{it} : IT attractiveness $\in [-1, 1]$
$t_{urgent} \leftarrow t_{due} - \gamma . (t_{due} - S_j)$;
$t_{late} \leftarrow t_{due} + \gamma . (t_{due} - S_j)$;
if $B_j \leq t_{urgent}$ **then**
  $a_{it} \leftarrow v_{base} + \alpha_{base} . \frac{t_{urgent} - B_j}{t_{urgent} - S_j}$;
else if $B_j \leq t_{due}$ **then**
  $a_{it} \leftarrow v_{urgent}$;
else if $B_j \leq t_{late}$ **then**
  $a_{it} \leftarrow v_{late}$;
else
  $a_{it} \leftarrow v_{bad}$;
end
Algorithm 3: Calculation of the IT attractiveness for a single possible task schedule.

Table 3
Values used for constants in Algorithm 3 to compute IT attractiveness.

α_{base}	v_{base}	v_{urgent}	v_{late}	v_{bad}	γ
0.2	0.7	0.2	−0.9	−1	0.1

and a starting time B_j. We will name thereafter a *placement* such a couple of a machine and starting time. Our algorithm, ABBSH (for Attractiveness-Based Blind Scheduling Heuristic), is online, scheduling tasks as and when they are submitted, without knowledge of the future submissions. It is implemented as a greedy heuristic, choosing a definitive placement for a task at the time it is submitted.

To decide this placement, the algorithm compares multiple possible placements. For each one, the IT attractiveness is computed, and the power consumption of the data center is evaluated using Eq. (1) by adding the resources use generated by this placement. The average power consumption is then used to make a request to the electrical loop, which gives the corresponding attractiveness as described in Section 3.4.1.

In order to chose among the possible placements, we are facing a multi-objective problem. Indeed, we would like to maximize

Function *placeTask(j, multiObjectiveFun)*
  placements ← empty list ;
  timeStep ← min($\alpha_{step}.T_j$, $step_{max}$) ;
  windowEnd ← D_j+ min($\alpha_{window}.(D_j - S_j)$, $window_{max}$) ;
  for t_{cur} ← *timeStep*; t_{cur} < *windowEnd*; t_{cur} ← t_{cur}+ *timeStep* **do**
    foreach *machine* ∈ $\mathcal{M}$ **do**
      t_s ← first time ≥ t_{cur} with at least (c_j, r_j) available on machine for a duration ≥ T_j ;
      $P_{required}$ ← average $P(t)$ between $[t_s, t_s + \frac{T_j}{C_m}]$, including current placement ;
      a_{elec} ← electrical attractiveness for proposal ($P_{required}$, t_s, T_j) ;
      a_{it} ← IT attractiveness for proposal (j, t_s);
      place ← (machine, t_s, a_{it}, a_{elec}) ;
      placements.append(place) ;
    end
  end
  (bestMachine, bestTime) ← multiObjectiveFun(placements) ;
  schedule j on bestMachine with B_j = bestTime ;

Algorithm 4: Simplified pseudo-code of the single task placement function of the ABBSH algorithm. The parameter *multiObjectiveFun* is a function which takes a list of placements with their attractiveness and returns the couple (time, machine) of the best solution.

both electrical attractiveness (which is an image of the *quality* of the power used) and IT attractiveness (representing the respect of SLA criteria). We decided to evaluate ABBSH using different multi-objective methods, which will be detailed in Section 3.6.

Function *onTasksSubmitted(tasks)*
  foreach j ∈ *tasks* **do**
    placeTask(j, multiObjectiveFun) ;
  end

`// called` $tboot_m$ `before` B_j
Function *onTaskReady(j, m)*
  if $p_m(t_{now}) = Off$ **then**
    startMachine(m) ;
  end

Function *onTaskStart(j, m)*
  executeTask(m, j) ;

Function *onTaskEnd(j, m)*
  t_{end} ← $t_{now} + \alpha_{reboot}(tboot_m + tshut_m)$;
  if $\forall_t \in [t_{now}, t_{end}], \mathcal{R}_m(t) = \emptyset$ **then**
    stopMachine(m) ;
  end

Algorithm 5: Simplified event-based scheduling and power management algorithm for ABBSH.

Algorithm 4 presents the method used to place a task j at its submission, using a given multi-objective function. The value of the time step used is function of the execution time of the task and of a coefficient, such as $\alpha_{step} \cdot T_j$. The time step is also bounded with a maximum, to have its value in [0, $step_{max}$]. Similarly, the length of the time window during which placements are considered is based on time between the submission and the due date, with another coefficient: $alpha_{window} \cdot (D_j - S_j)$. To keep this duration into a known interval, a maximum value is also used, keeping the time window in [0, $window_{max}$]. The values of those parameters as used in our experiments will be given in Section 4.4.

To reduce the global power consumption of the data center, our approach turns off the machines when they are unused. By having the usage planning of all computing resources in the near future, it is easy to anticipate the necessary boot-up. The overall scheduling and power management algorithm is presented in Algorithm 5, in an event-driven form. Then functions *onTasksSubmitted* and *onTaskStart* implement the scheduler itself. The power state of the machines is managed by *onTaskEnd*, which allows to turn off idle machines, and *onTaskReady*. The latter is called before the scheduled execution of a task, to let the time to turn on the corresponding machine if needed. Turning on and off the machines too often may result in some drawbacks, including an increase of the total consumption, because of the additional energy required during those steps. In order to avoid unnecessary reboot, Villebonnet et al. [34] define a "minimum switching interval", which is the duration under which it is preferable to keep the machine idle than to do a complete reboot cycle. In our scheduling algorithm, the parameter α_{reboot} is used for a similar purpose, but is relative to the time required for a reboot cycle. A machine is switched off only if it is expected to be idle for more than $\alpha_{reboot} \cdot (tboot_m + tshut_m)$, with $\alpha_{reboot} > 1$.

3.6. Multi-objective methods

As our algorithm requires a multi-objective function to select the best placement for each task, we decided to evaluate our approach using different of such methods. The objectives are, as mentioned in Algorithm 4, IT and electrical attractiveness. Three multi-objective methods were tested. The first is a weighted sum of both objectives, which is a simple method commonly used in the literature. The second consists of a weighted sum of the hyperbolic sinus of both attractiveness. Finally, the third is a variant of the *fuzzy-based* approach presented by Sun et al. [35].

The use of a weighted sum is particularly interesting here, as the attractiveness values are already normalized. The choice of the hyperbolic sinus variant is motivated by the shape of this function. In addition to its symmetry, due to the fact it is an odd function, it also has interesting properties which give more weight to the extreme attractiveness values (close to −1 or 1). These functions are defined in Eq. (4a) for the simple weighted sum and Eq. (4b) for the sum of hyperbolic sinus. The weighting is controlled by a parameter $\alpha \in [0, 1]$. Parameter β in Eq. (4b) is used to control the shape of the hyperbolic sinus transformation. A value close to 0 makes the transformation almost linear in [−1, 1], and the greater β is, the more the weight of the extreme attractiveness values are important compared to medium ones.

$$f_{wsum}(a_{it}, a_{elec}) = \alpha \cdot a_{it} + (1 - \alpha) \cdot a_{elec} \tag{4a}$$

$$f_{wsinh}(a_{it}, a_{elec}) = \alpha \cdot \sinh(\beta \cdot a_{it}) + (1 - \alpha) \cdot \sinh(\beta \cdot a_{elec}). \tag{4b}$$

The *fuzzy-based* method allows to select a solution in the Pareto front, which is the set of non-dominated possible solutions. Fig. 4 illustrates this method, with a scenario where two objectives are

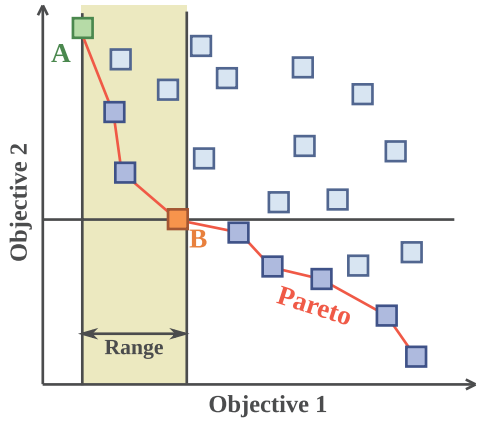


Fig. 4. Illustration of the *fuzzy-based* multi-objective approach, with two objectives to minimize.

to be minimized. Initially, the best solution for the first objective is selected (point A in the figure). A margin is added to this minimum value of the first objective, and all the solutions in this range are considered. The final solution is the one, inside this subset, which minimizes the second objective (B in Fig. 4). By choosing a value for this margin, it is possible to adjust the constraint for the optimization of the first objective. In our implementation, the margin used is relative to the minimum and maximum values of the first objective, and to a parameter called fuzzy factor (named α for uniformity with the other methods). With min_{obj1} and max_{obj1} the extreme values of the solutions on the first objective, the margin is given by $\alpha(max_{obj1} - min_{obj1})$.

4. Methodology

Our approach has been developed and evaluated in a simulated platform, and compared to the GreenSlot scheduling algorithm [12]. After describing the simulation environment, we will detail the workload generator we used to simulate user jobs submission. Then, we propose to present briefly the original GreenSlot scheduler and the modifications we added either to adapt it to our model and to improve it for the considered workload. Finally, we will detail the parameters used in our experiments for instantiating the models presented in Section 3.3.

4.1. Simulation platform

Our simulation platform is based on DCworms [36], a data center simulator designed mainly for studying the power consumption in large-scale systems. This simulator was already used in several works focused on the impact of scheduling policies on a data center consumption [35,37]. However, as far as we know, this is the first time it is used in a renewable energy context.

To be able to take decisions based on availability of renewable energy, we extended DCworms by implementing electrical infrastructure simulation. The infrastructure is represented as a set of interconnected nodes, each of them being an electrical component, such as a renewable source, a grid supply, an electrical bus or a converter. Each node implements the element's model as a DCworms plugin, written in Java. The whole electrical infrastructure description is provided to the simulator along with the computing one, as an XML dialect input file.

For the purpose of the presented experiments, we used a simple infrastructure, composed of solar panels and of a grid supply which simulates price variation during the day. The solar panel power production model approximates the solar irradiance using a half-sinus with a period of 24 h, having its peak centered at noon.

Listing 1: Simplified python code for generating each task of the workload, resulting of the application of our parameters to the model from [38].

```
from scipy.stats import (pareto,lognorm,expon)

def truncated_expon(f):
while True:
  val = expon.rvs(scale=1.0/f)
  if val <= 1.0:
    return val

def get_next_task(prevSubmission):
arrival = pareto.rvs(4, loc=1) * 3 * 72
submission = prevSubmission + arrival
priority = truncated_expon(6)
makespan = lognorm.rvs(1.634, scale=447)

return (submission, priority, makespan)
```

4.2. Workload

The workload used to simulate the user jobs submission is a crucial point when studying schedulers. Depending on the targeted data center usage, the real workloads differ a lot. We decided to generate a synthetic workload based on a previous study of a Google large-scale cloud computing cluster [38]. By using such workload generator, we can easily control its duration, and generate several workloads based on the same distribution laws, but using different random seeds.

The model from [38] allows to generate tasks and services of different *priority levels* and execution time, controlled with few parameters. As our work is only focused on batch tasks, we set the parameter $ratioTask = 1.0$. We used the same parameters related to execution time as the ones extracted from Google traces in the article, which are $mass = 1700$ (average task execution time in seconds) and $disparity = 3.8$ (ratio between average and median execution time). The value of *dynamism*, which is the average inter-arrival time, has been adapted to fit the size of our simulated computing infrastructure, and we used a value of 72 s.

With those parameters, we can extract both inter-arrival and execution time distributions from the model. The Python code in Listing 1 gives the distributions used and their parameters using the SciPy package [39]. The inter-arrival time follows a modified Pareto law, which is scaled according to *dynamism*. The execution time, named *makespan* in the code, is modeled by a log-normal distribution, for which the parameters are calculated to respect a mean equals to *mass* and a median of *mass*/*disparity*. Finally, the *priority* of the task, which is used to determine its due date, is given by an exponential distribution of rate 6, truncated to keep the value between 0 and 1.

Unfortunately, this model does not handle the tasks due dates. As we have no knowledge of any work on the modeling of such due dates in cloud workloads, we propose an arbitrary model based on tasks priorities. It is parameterized with a value named *flexibility factor*, which is used to modify the magnitude of individual tasks flexibility (time available for scheduling the task with respect to its due date, as defined in Section 3.4.2). The impact of this factor on the results of the different heuristics will be pointed out in Section 5.

Defining such a metric and studying how it affects the behavior of this kind of scheduling algorithm seems critical for us. Indeed, it is obvious that the performance of algorithms based on delaying tasks to execute them at the best time (here, when the renewable sources produce enough power) depends directly on *how much* those tasks can be delayed. Because of the lack of existing study, we cannot use known work to take realistic distributions of the

Table 4
Parameters of truncated normal distribution used to define task flexibility (in seconds) depending on its priority class.

Priority class	μ	σ
Low	3600	600
Normal	1200	300
High	0	0

flexibility depending, for example, on the kind of data center or the politic of its operator.

Based on this observation, two possibilities exist. The first, used by most of the related study, is to use a workload with arbitrary due dates and analyze the proposed approach on it. The second one, which seems preferable for us, is to propose an arbitrary, parameterized distribution of the tasks flexibility and to confront the approach to different instances of this distribution. The results are therefore easier to extrapolate to real workloads (by measuring the distribution of the due dates). In addition, they provide a better overview of the performances of an approach, for a wider range of situations. For those reasons, we strongly believe that defining and studying a flexibility factor results in more interesting comparisons between different approaches.

Let *flexf* be the flexibility factor used. The due date D_j for a given task j is obtained by Eq. (5), where f_{min} is a constant used to define a minimum flexibility and f_{base} is the *base* flexibility. For our needs, the minimum flexibility value is set to $f_{min} = 60$ s, and the base flexibility is obtained using random distribution law that depends on the task priority.

$$D_j = S_j + f_{base} \cdot flexf + f_{min}. \tag{5}$$

Using the workload model from [38], each task has a priority as a value $\in [0, 1]$. Based on that value, we determine a *priority class*, which is either *low* (for priority between 0 and 1/3), *normal* (between 1/3 and 2/3) or *high*. The value of f_{base} is then obtained with a normal distribution, truncated at $\mu \pm 3\sigma$, with the values of μ and σ defined for each priority class, as given in Table 4. For the priority class *high*, tasks have only the minimum flexibility, resulting in μ and σ both equal to 0.

In our experiments, each workload starts at midnight, and contains tasks submitted during 72 h (corresponding to an average of 3600 tasks). As the log-normal distribution used for execution time is long-tailed, some very high values may occasionally be assigned to T_j. The time period considered for the workload being relatively short, we decided to only accept tasks with $T_j \leq 24$ h (if a task is longer, a new task is generated instead).

4.3. GreenSlot

We implemented the "GreenVarPrice" version of the GreenSlot scheduler, which takes into account the variation of grid energy price, based on its description given in [12]. Several simplifications were made in the algorithm presented in the aforementioned paper, along with some modifications to consider the same machine and task model:

- As we have a perfect renewable prediction, we removed the adjustment and re-scheduling in case of prediction error.
- The task execution time is perfectly known, and we do not add tolerance to the time specified at submission.
- Instead of rejecting a job at submission if we cannot respect its due date, we accept it and schedule it after its due date.
- In our implementation, a machine may execute several tasks at the same time (depending on the number of processors installed and required), whereas the original study considered tasks using the whole machines.
- In the original article, the idle nodes were put in sleep mode (S3 state). Instead, our implementation turn them off, based on the model presented Section 3.3.1.

This algorithm is based on a discrete time unit (15 min in the original paper experiments), named time *slot*. It considers a given time window, divided into slots. For each slot, the algorithm keeps the prediction of renewable energy production at that time, the available computing resources, the price of grid energy, and the planned consumption during this slot.

The tasks submitted by the users are put in a waiting queue in Least Slack Time First (LSTF) order. At the beginning of each time slot, the waiting tasks are scheduled in LSTF order. For each task, the algorithm computes a cost for each slot in the window, corresponding to the cost for starting the task at the beginning of this slot. This cost for a given slot is infinite if the required computing resources are not available during all the task execution time, or if the task is expected to end outside the current window. Otherwise, the slot cost depends on the total cost of the grid energy required, plus an eventual penalty if starting the task at this time violates its due date.

Eq. (6a) gives the cost function, for a given couple of starting slot t_s and task j, when it is not infinite. The set of time slots during which the task is executed is given by S. For a slot s, we have $gridEnergy_s$ the amount of energy which will be required from the grid at this time (in kWh) and $gridPrice_s$ the price of grid energy for this slot (in \$/kWh). A coefficient c_{grid} is used to obtain the cost considered by the algorithm. Finally, an additional *penalty* may be added to the total cost, as given by Eq. (6b). This value can be either 0 or $penalty_{violated}$, depending if the due date is respected or not. The values used as c_{grid} and $penalty_{violated}$ are not given in the original work, but we assume the due date violation to be important compared to some grid energy use. We used $penalty_{violated} = 5$, and $c_{grid} = 1$ in our experiments.

$$penalty + \sum_{s \in S} c_{price} \cdot gridPrice_s \cdot gridEnergy_s \tag{6a}$$

$$penalty = \begin{cases} 0 & \text{if } t_s \leq D_j - T_j \\ penalty_{violated} & \text{otherwise.} \end{cases} \tag{6b}$$

When the costs, for a given task, are calculated for every possible starting slot, it is scheduled to begin at the slot with the lowest cost (choosing the earliest if several have the same cost). Once scheduled, the available computing resources and planned consumption are updated for the slots during which the task will be executed.

However, during our early experiments with the GreenSlot algorithm, we found it to be poorly suited to the kind of workload we used. The one used in the original work [12] was a scientific computation workload, with few short tasks and due dates relatively far away from the submission. Our workloads, at the contrary, contain a lot of short tasks, and some of them have to be started in a minute.

We identified two causes of bad performances when using our workloads with the original algorithm. Appropriate modifications are proposed to improve it in order to adapt it to our use case.

1. A task with a low flexibility, submitted in the middle of a slot, may need to be scheduled before the next slot to have its due date respected. As the task queue is processed only at the beginning of each slot, such a task is likely to have its due date violated.
2. As the original algorithm considers a time slot to be indivisible, a task which uses only a small part of a slot (either a very short task or the last slot in which a longer task is running) will appear to reserve an entire slot. Because of that, the computing resources usage become fragmented, causing the algorithm to need more resources than really required, or even to delay tasks after their due dates because the nearest slots are already reserved.

Table 5
Instantiation of the IT model described in Section 3.3.1 (homogeneous data center, the values are given $\forall m$).

Parameters	Meaning	Values
M	Number of machines	10
N_m	Processors per machine	4
R_m	Memory per machine	32 GiB
ps_m	Base machine consumption	44 W
$pmin_m$	Processor consumption at idle	0 W
$pmax_m$	Processor consumption at full load	21.5 W
C_m	Processor relative speedup	1
$pboot_m$	Total power used during boot up	120 W
$tboot_m$	Time for booting up	40 s
$pshut_m$	Total power used during shut down	100 W
$tshut_m$	Time for shutting down	15 s

To address the first issue, we propose to modify the task submission processing, to schedule it immediately if its due date will be violated before the next slot, instead to put it in the waiting queue. For the second issue, we improve the algorithm to make it able to reserve resources for a fraction of a slot. It is then possible to *backfill* a slot with other tasks, starting immediately after the previous one.

Because of those modifications, we used three different versions of the GreenSlot algorithm in our experiments. The first, referred as *original*, contains none of those modifications, and uses a slot time of 15 min (the same value used in the experiments of the authors of the GreenSlot approach). The second, named *partially modified*, uses a shorter time slot (5 min) and implements the urgent scheduling modification. Finally, the *modified* version implements both modifications, and a time slot of 15 min, as the second modification reduces the need of short time slot. The performances of those different versions will be detailed in Section 5.2, allowing to argue about the two issues and the impact of our modifications in the GreenSlot algorithm.

4.4. Experimental configuration

Before to present the experiment results, we will detail the model instance we used and the values of the various parameters mentioned previously.

IT model. We simulated a small data center, quite similar in peak consumption and computing power to the one used by the authors of [12]. To use realistic values of power consumption, we chose an energy-efficient commercial rack server as a reference. It is the Dell PowerEdge R210 II, with a 4-cores Intel Xeon E3-1200 series processor and 32 GB of RAM.

The consumption of this server is measured up to 130 W at peak load and 44 W at idle.[4] This is slightly higher than the values announced by the manufacturer in its power and performance data sheet, but in accordance to other measurement on comparable servers [40]. Our consumption model only takes into account the dynamic consumption of the processors, so we will attribute to them all the power difference between idle and full load. The time required for turning on the server was not measured directly for this model. Very few works studied in detail the boot time of servers. There is a wide range of measured values depending the architecture and the software configuration [41], as shown for instance by Villebonnet et al. [40] measuring from 12 to 189 s. A value of 40 s seems reasonable for a headless server, as the measurements from [41] are comparable in the case of several machines with GNU/Linux distributions including a graphical session start. The measurements are even lower (around 30 s) when booting from network instead of using a hard drive for this purpose. To take into account the peak of power consumption which occurs when turning on and off a machine [42], we considered a consumption close to the full load one.

The simulated data center is composed of 10 of those servers, or 40 processors available. The model from Section 3.3.1 is therefore instantiated with 10 identical machines with values summarized in Table 5.

Electrical model and attractiveness. The electrical infrastructure is, as mentioned in Section 4.1, composed of two electrical sources. The first one is a set of solar panels, providing a peak production of 1500 W. Depending of the photovoltaic module used, the cells technology, and the efficiency of the solar inverter, the required surface of solar panels may vary a lot. We will consider an efficiency of around 15%, which is a conservative value of what is commonly found in the current market, for modules using single or multi-crystalline technologies cells [43]. With such efficiency, a peak production of 1500 W is achievable with 10 m^2 of panels, assuming a peak solar irradiance of 1000 $W.m^{-2}$ [44]. The time window for prediction of energy availability and grid energy pricing, $window_{prediction}$ is set to 48 h. As a matter of simplification, and as the electrical infrastructure is implemented in a simulator, the prediction model used here is an oracle, with perfect prediction during the whole window ($predicted_{solar}(t) = available_{solar}(t)$). The second source is the electrical grid, with an on-peak/off-peak pricing based on the values used in [12]. Two periods are considered for pricing: \$0.13/kWh from 9 am to 11 pm (on-peak), and \$0.08/kWh from 11 pm to 9 am (off-peak).

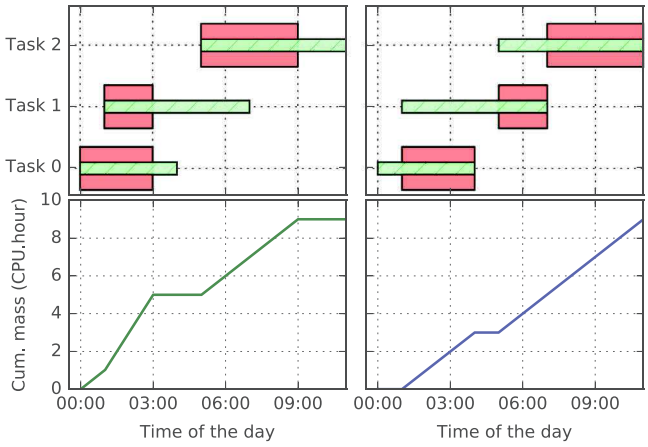
Having the maximum renewable sources production and the grid prices, we can define the missing constants used in electrical attractiveness functions. The values of h_{pos} and h_{neg} determine the rate at which attractiveness increase or decrease depending on the available power. We set both to a quarter of the peak renewable power, $h_{pos} = h_{neg} = 375\,\mathrm{W}$. For the variant B of the attractiveness function, we have the minimum and maximum grid energy price, $\theta_{min} = 0.08$ and $\theta_{max} = 0.13$. The *price factor* λ will be set by experiments detailed in Section 5.1.

Scheduling algorithm. Several parameters are used in the task placement algorithm itself (Algorithm 4). Two couples of parameters, related to time steps (α_{step} and $step_{max}$) and to the time window considered for scheduling (α_{window} and $window_{max}$) are used by the algorithm, as described in Section 3.5. A last parameter, α_{reboot}, control the aggressiveness of the shutting down of idle machines. The values used for the first four parameters, presented in Table 6, were chosen based on preliminary experiments for giving good results for both short and very long tasks. However, they were not tuned specifically for the workloads used in this paper and different values may lead to better results, at the cost of a longer execution time of the task scheduling algorithm.

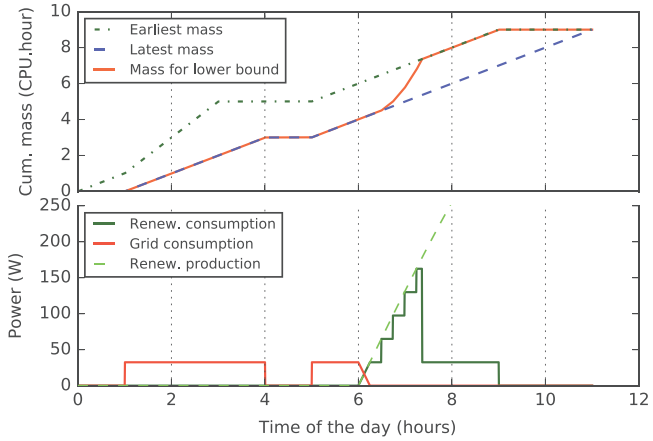
For the parameters related to the time window, the chosen values allow the scheduling of tasks long after their due dates (up to 4 times the duration of the task with a maximum of 12 h). This permits possible SLA violations if a heuristic considers it more interesting than using more energy at some point to respect the due date of a task. The ones related to the time steps are chosen to avoid too long steps for long tasks (the maximum time step is 30 min), while keeping steps long enough for short tasks (0.3 times its normalized execution time T_j).

Finally, choosing $\alpha_{reboot} = 2$ allows to keep a machine turned on if it is idle for less than twice the time needed to perform a complete reboot. Choosing higher value results in more idle machines not powered off and a lower value results in more aggressive reboot of machines to avoid any idle time. As the machines consume more power when shutting down or booting up than in idle state, this value may also be tuned with the characteristics of the machines used. For sake of completeness, the optimal in

[4] Power consumption using SiSoft Sandra benchmark, as measured by http://www.itpro.co.uk/635800/dell-poweredge-r210-ii-review.



(a) Tasks in workload and corresponding cumulated computing mass, when executed at earliest (left) and at latest (right).



(b) Solution of the lower bound method for this workload, with associated consumption across time.

Fig. 5. Illustration of the method for finding the lower bound of grid energy consumption.

Table 6
Values of parameters used in the scheduling algorithm.

α_{step}	$step_{max}$	α_{window}	$window_{max}$	α_{reboot}
0.3	0.5 h	4	12 h	2

terms of power consumption and for an homogeneous data center is given by Eq. (7). This equation gives the value of α_{reboot} for which the total energy consumed during a reboot cycle is equal to the energy consumed if the machine stayed idle.

$$\alpha_{reboot}^{optimal} = \frac{pboot_m tboot_m + pshut_m tshut_m}{(ps_m + pmin_m N_m)(tshut_m + tboot_m)}. \tag{7}$$

Workloads. The workloads are generated with the model described in Section 4.2, which gives the values, for each task j, of the parameters S_j (submission date), D_j (its due date) and T_j (its execution time). Several values of flexibility factors *flexf* are used, depending on the experiments, and will be given along with the results. In order to keep the workload definition simple enough, all the tasks have the same CPU and memory requirements, $n_j = 1$ and $r_j = 1$ GiB. In our experiments, the measured average load of the computing infrastructure with such a workload is about 45%, which is higher than most of the typical data centers [45], and similar to some modern cloud clusters.

To ensure the validity of our experiments, we generated a set of 10 different workloads for each flexibility factor used. Those 10 workloads are generated with the same probability distributions, but using different random seeds. All the results given in the next sections were repeated on the 10 workload instances, and the values used are the average, unless specified otherwise.

4.5. Lower bound of grid consumption

The methodology presented here was focused on comparing our approach to others heuristics from the literature. However, it does not give hints to know how close those results are from the optimal. Scheduling problems are, unfortunately, known to be easily intractable, even for simple models [46]. In the particular problem presented in this paper, we expect that obtaining an optimal solution (thanks to linear programming for instance) within a reasonable time can be achieved only on tiny instances. Therefore, we propose instead to study a lower bound of the energy consumed on the grid.

The main idea is to relax the problem in a continuous way, such that individual tasks are gathered into a fully parallelizable workload (described as a *mass*). This mass can be assigned indistinctly onto the machines, while keeping information about submission and due dates of the tasks. We keep the notations of the model presented in Section 3.3 and relies on the following assumptions:

- A task may be executed on any number of processing unit (fully parallelizable tasks), suspended and migrated instantaneously with no cost.
- The data center is homogeneous (the subscript m, used in the notations from Section 3.3.1, will refer to any machine and their speedup $C_m = 1$).
- The machines can be powered on and off instantly.
- The power consumed by a machine is proportional to the amount of processor used (each accounting for $ps_m/N_m + pmax_m$).

As the problem has been relaxed with these assumptions, we should note that the lower bound would be unachievable.

We focus on two characteristics of the workload. The first one, $Mass_{earliest}$, given by Eq. (8c), is the cumulated mass if every task is scheduled as early as possible. It is easy to compute based on $Wl_{earliest}$, the mass given in Eq. (8a). Eq. (8d) gives $Mass_{latest}$, the second one, when the tasks are scheduled as late as possible while respecting their due dates, based on the mass from Eq. (8b).

$$Wl_{earliest}(t) = \sum_j \begin{cases} n_j & \text{if } S_j \le t \wedge S_j + T_j \ge t \\ 0 & \text{otherwise} \end{cases} \tag{8a}$$

$$Wl_{latest}(t) = \sum_j \begin{cases} n_j & \text{if } D_j - T_j \le t \wedge D_j \ge t \\ 0 & \text{otherwise} \end{cases} \tag{8b}$$

$$Mass_{earliest}(t) = \int_0^t Wl_{earliest}(x)\,dx \tag{8c}$$

$$Mass_{latest}(t) = \int_0^t Wl_{latest}(x)\,dx \tag{8d}$$

$$\forall t \quad 0 \le Cpu(t) \le N_m \cdot M \tag{8e}$$

$$Cpu(t) \ge Mass_{latest}(t) - Mass_{done}(t-1) \tag{8f}$$

$$Cpu(t) \le Mass_{earliest}(t) - Mass_{done}(t-1) \tag{8g}$$

$$MassDone(t) = \int_0^t Cpu(x)\,dx \tag{8h}$$

$$Cons(t) = Cpu(t) \cdot \left(\frac{ps_m}{N_m} + pmax_m\right) \tag{8i}$$

$$Grid(t) = \max(0, Cons(t) - \sum_{w}^{W} available_w(t)). \quad (8j)$$

Fig. 5(a) illustrates that using an example workload with only 3 tasks. The tasks are represented in the top figures, along with the period between their submission and their due dates (in green). The cumulated mass associated to the earliest and latest scheduling are given respectively in the bottom left and bottom right subfigures. Any valid scheduling which respects all the due dates must have, at each time, its cumulated mass between $Mass_{latest}$ and $Mass_{earliest}$. This is represented by the constraints in Eqs. (8e) to (8h). $Cpu(t)$ is the total amount of processors used at a given time t, while $MassDone(t)$ is the associated cumulated mass.

Based on these constraints, it is possible to find a lower bound of the energy consumed from the electrical grid. Eq. (8i) gives the consumption of the data center, $Cons(t)$. By removing the power produced by each of the W renewable sources, $Grid(t)$ represents only the power coming from the grid, given by Eq. (8j). An illustration of the method, for the example workload, is shown in Fig. 5(b). At the top, the earliest and latest cumulated mass are represented, as well as the one found as the lower bound. The resulting consumption, for both grid and renewable, are given in the bottom subfigure.

In practice, we used a discrete time model, with time steps small enough, to compute the lower bound. The solution is therefore given by minimizing $\sum_t Grid(t)$. At each time step, the renewable power is used to do as much work as possible without exceeding the earliest cumulated mass. Then, if it was not enough to reach the latest cumulated mass, grid power is used for this purpose.

5. Results

In order to present our results, we will, in a first time, present the experiments we used to fix the parameters of the several multi-objective methods detailed in Section 3.6, and the *price factor* used in variant B of the electrical attractiveness function. Then, using the values determined by those experiments, we will compare the performances of the different approaches and heuristics on several criteria, and for multiple *flexibility factors* of the workload.

5.1. Choosing parameters

Multi-objective functions parameters. We have a total of four different multi-objective methods that may be used by ABBSH: the simple weighted sum, the weighted sum of hyperbolic sinus, the *fuzzy-based* with IT attractiveness as its first objective (fuzzy-IT) and the one with electrical attractiveness as its first objective (fuzzy-elec). For each of them, we have an α parameter which controls its behavior. For the two weighted sums, it is the weighting factor, and it is the fuzzy factor used to calculate the margin on the first objective for the fuzzy-based approach. The parameter β, used for the weighted hyperbolic sinus sums, was fixed to a value high enough to give more weight to the extreme attractiveness value, and we used $\beta = 2.5$.

The value used impacts directly the preference toward either the IT or the electrical attractiveness. Ideal value depends on the formulation of attractiveness function. As these formulations do not change during the use of the scheduling algorithm, choosing the value is expected to be done offline and only once for configuring the scheduler.

To choose a value for them, we used workloads generated with a relatively high flexibility ($flexf = 8$), and used only the electrical attractiveness variant A. The experiments consist in changing the α value, and evaluating the impact of it on two criteria: the part of non-renewable energy used and the rate of tasks that finished their execution after their due dates. The results are shown in Fig. 6. We are looking for a value of α, for each method, which minimizes both criteria.

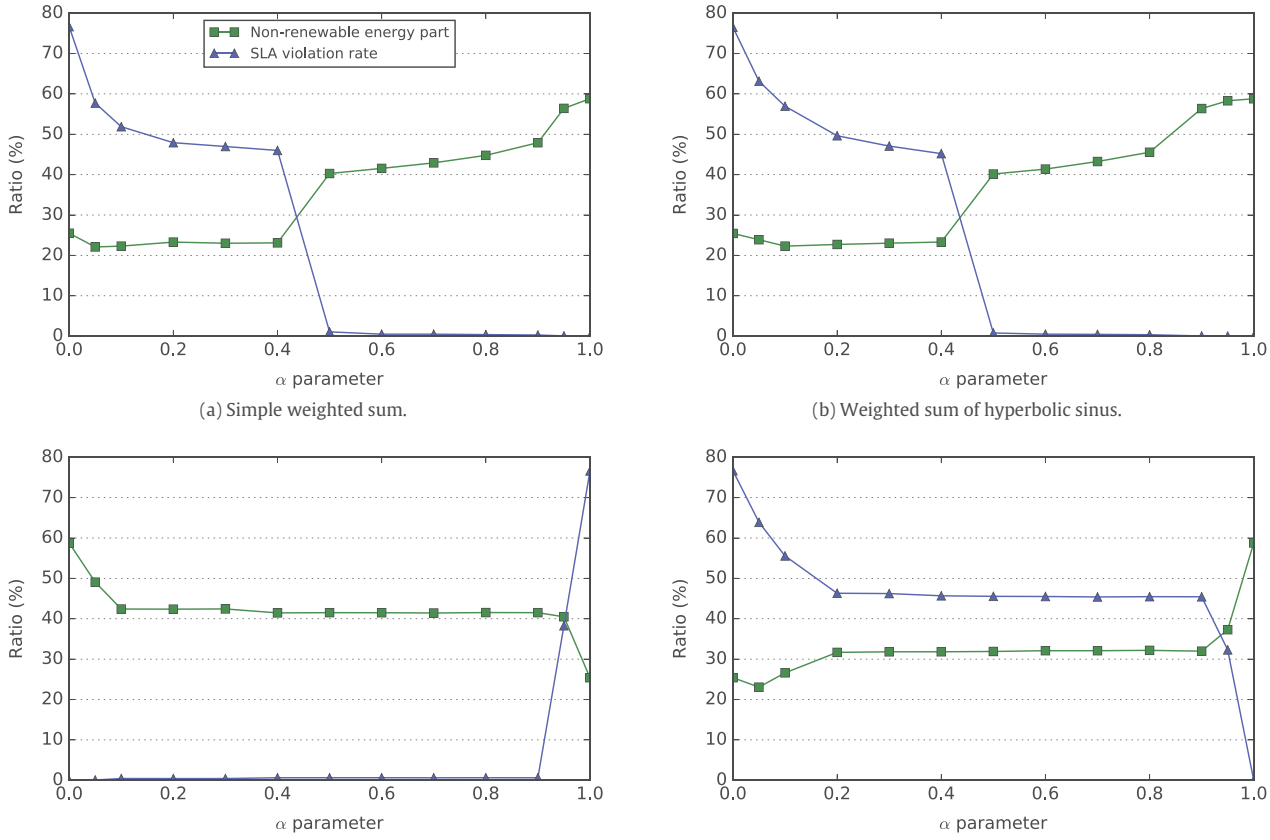
For the weighted sums, the results in Figs. 6(a) and 6(b) are quite similar. The SLA violation rate decreases quickly when α becomes higher than 0.5, and the non-renewable part increase when α is close to 1. To keep a high QoS and a high renewable energy usage in the same time, we choose $\alpha = 0.55$ for both functions.

For the fuzzy-based approaches, a first look to Fig. 6(d) indicates its worthlessness when the electrical attractiveness is the first objective. Indeed, the margin needed to have a SLA violation rate near zero seems to imply a less important renewable usage than for the other methods. For this reason, we will not keep it in the subsequent experiments. Fig. 6(c) shows a very different result when the first objective is the IT attractiveness. In this case, we note that non-renewable energy usage decreases a little bit when α increases between 0.1 and 0.9, staying almost constant. Similarly, the SLA violation rate is slowly increasing, until it grows quickly when α reaches 0.9. We will choose a relatively high flexibility factor of $\alpha = 0.75$, to keep the renewable energy usage as important as possible, and without risking to degrade too much the QoS.

Electrical attractiveness variant B. With the multi-objective parameters fixed in the case of the variant A of the electrical attractiveness function (without taking the electricity price into account), we can study how to integrate the price in the variant B. The influence of the electricity price from the grid is controlled by λ, the *price factor*. The choice of its value is to be considered as a decision of the data center operator. It fixes the trade-off between privileging the renewable energy (low value) or the cheap grid energy (high value). With $\lambda = 0$, the behavior is the same as with the variant A (we find $base = \alpha_{neg} = -0.7$), guaranteeing that the parameters chosen for the multi-objective methods will work as well for this case of the variant B. By increasing it, the attractiveness value, for a situation where renewable energy is not sufficient, will increase compared to the variant A by at most λ (in the case of the cheapest possible energy prices).

Fig. 7 gives the results of non-renewable energy consumed (bought from the electrical grid) and its total cost. The impact of λ is similar for the different multi-objectives methods. The total cost decreases initially, but starts increasing again when the price factor becomes too high (≥ 1.3). This behavior is related to the total grid energy consumed. Indeed, the total cost is correlated with the energy consumed from the grid, and the average price for this energy. With small values of λ, the grid consumption increases a bit compared to $\lambda = 0$, but as the price is taken into consideration, the average price decreases in the same time. However, when λ becomes high enough, increasing it does not reduce significantly the average energy price, and the total price simply follows the grid consumption. Worst, when the price factor is too high, the electrical attractiveness function tends to favor the use of cheap grid energy instead of renewable one which covers only partially the requirement. Consequently, the amount of grid energy increases again and the total cost follows. Those high values are probably not the behavior wanted by any operator of data center with renewable energy sources, but shows how important it is to choose an appropriate value for λ.

With those results, we can choose a value for the price factor, used by the three different heuristics. We want a value that increases the least possible the non renewable energy consumption, while decreasing the grid energy cost the most possible. Therefore, setting the price factor $\lambda = 1.2$ seems reasonable, and will be used for the next experiments. With this value, compared to the variant A, the weighted sum with the variant B uses 2.2% more grid energy but reduces the cost by 12%. Similarly, with the weighted sum of hyperbolic sinus, the consumption from the grid is 0.7% higher for a cost saving of 13%, and respectively 1.0% and 4.9% for fuzzy-IT.



(a) Simple weighted sum.

(b) Weighted sum of hyperbolic sinus.

(c) Fuzzy-based with IT attractiveness as first objective.

(d) Fuzzy-based with electrical attractiveness as first objective.

Fig. 6. Impact of multi-objective functions parameter value (weighting or fuzzy margin) on SLA violation and on part of grid energy used.

5.2. Impact of the workload flexibility

With the multi-objective functions parameters set and electrical attractiveness well defined, we can compare the performance of the different heuristics depending on the flexibility factor used to generate the workload. Greater flexibility factor gives tasks with more time between their submission and their due dates and therefore more opportunity for a scheduling algorithm to delay them to a better time while respecting their SLA. We used 6 different values for the flexibility factor *flexf* : 0 (resulting in tasks needing to be executed immediately after their submission), 1, 2, 4, 8 and 16.

For each value of *flexf*, 10 workloads were generated, using the same parameters described in Section 4.2 but with different random seeds. We evaluated 9 different heuristics which are the following.

- the 3 GreenSlot variants (original, with partial modifications and fully modified)
- ABBSH with electrical attractiveness function A in combination with 3 multi-objective methods (weighted sum, weighted sum of hyperbolic sinuses and fuzzy-based meth od with IT attractiveness as first objective)
- ABBSH with electrical attractiveness function B in combination with the same 3 multi-objective methods.

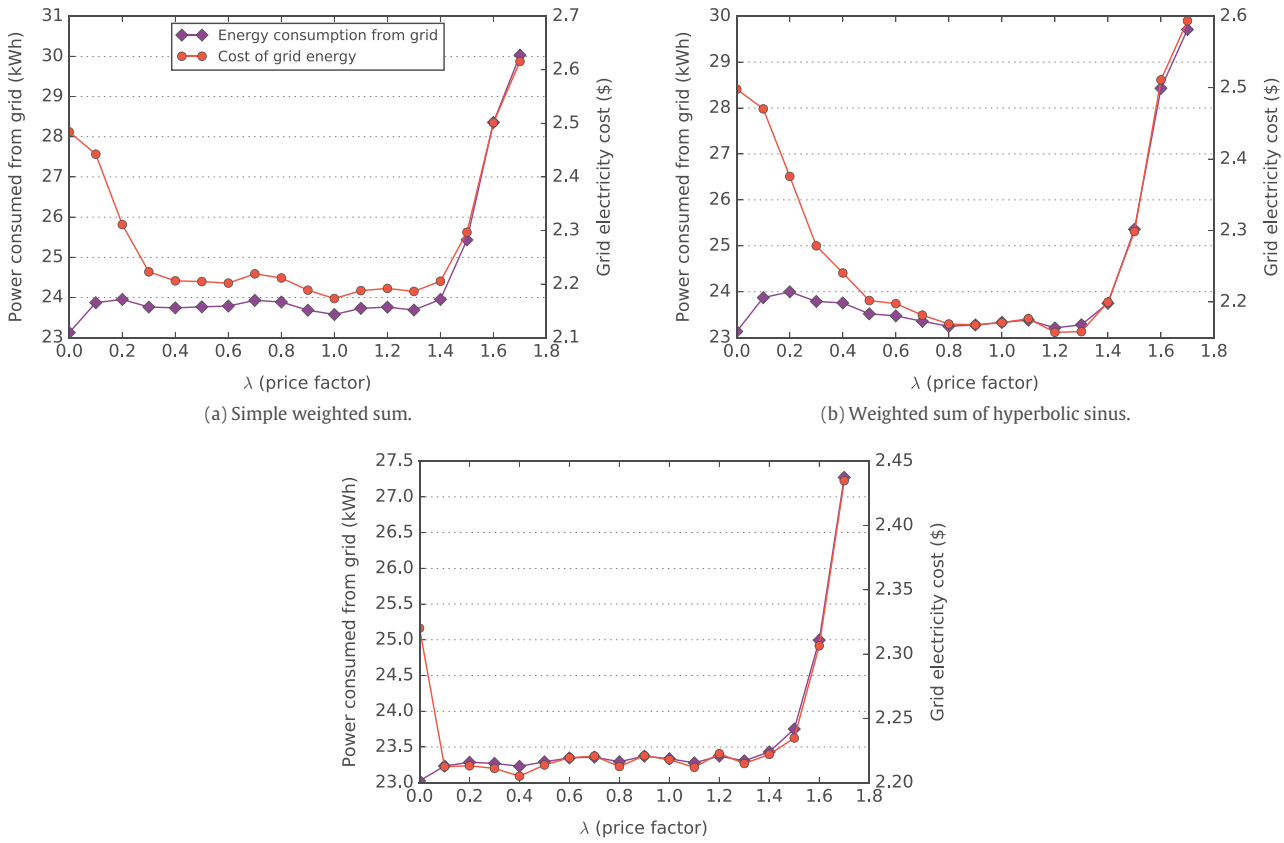
These 9 heuristics were each applied to all of the 10 workloads of each value of *flexf*. Several metrics were measured (percentage of tasks for which SLA was violated, amount of grid energy consumed and total price of this grid energy) to compare the different heuristics. During the rest of the section and unless stated otherwise, the results presented are the average value over the execution with the 10 workloads for a given combination of heuristic and flexibility factor.

GreenSlot versions. Fig. 8 gives the results for the three GreenSlot versions. We note the high SLA violation rate for the *original* version, in Fig. 8(a), which reaches 98% when the workload flexibility factor is 0. The violation rate is less than 2.5% for higher flexibility values, but is still several times more important than with the two other versions, which include our modifications. The same figure shows that the violation rate for the *partially modified* and *modified* versions is increasing with the flexibility factor.

Fig. 8(b) presents the power consumption from the grid for the different GreenSlot versions. Each tends to consume less grid energy when the flexibility factor increases. The *original* version consumes always more than the two others, for the reasons given in Section 4.3. The cost of the grid energy, given in Fig. 8(c), is very similar to the amount of energy drawn from the grid. However, we can notably note that the cost is not exactly proportional to the energy consumed. The results suggest that GreenSlot scheduler use cheaper energy when the flexibility increases, as we can expect from a price-aware algorithm.

We can see how our modifications allow to improve the performances of the GreenSlot approach when used with our kind of workload, composed in large proportion of short tasks. As the *modified* version is better in all criteria compared to the others, we will use it as a reference when presenting the heuristics of our approach.

Electrical attractiveness variant A. Fig. 9 presents the results obtained with the several multi-objective methods of ABBSH, using the variant A of the electrical attractiveness function (no grid



(a) Simple weighted sum.

(b) Weighted sum of hyperbolic sinus.

(c) Fuzzy-based with IT attractiveness as first objective.

Fig. 7. Impact of *price factor* variation on the amount of energy bought from the electrical grid and its total cost (with the variant B of electrical attractiveness function, which considers the price variations).

energy price consideration) and the GreenSlot *modified* version. The SLA violation rate (Fig. 9(a)) has a similar shape for all the heuristics, increasing with the flexibility factor. Our approach leads to a bit more violations compared to GreenSlot, especially when the flexibility factor is high.

The consumption of non-renewable energy, shown in Fig. 9(b), is decreasing when the flexibility factor increases for all the heuristics. The fuzzy-IT method gives better results than the others when the flexibility factor is low ($\leq$4 at least) or high (16), but is worst for medium value. Both simple weighted sum and the weighted sum of hyperbolic sinus give almost identical results, always equal or worst than the modified GreenSlot.

As the total cost for the grid energy is correlated with the amount of grid energy used, the behavior of the different heuristics is similar for this metric (Fig. 9(c)). We can note, however, that GreenSlot takes advantage of the tasks flexibility to use cheap energy, which allows to reduce the cost more efficiently with high flexibility values than our approach with the variant A of electrical attractiveness. In addition, the fuzzy-IT method of our approach seems to use cheaper energy than the two others, without the grid price being considered by the heuristic.

Electrical attractiveness variant B. The same metrics are presented, for the variant B of our electrical attractiveness function, in Fig. 10. SLA violation rate, in Fig. 10(a), follows a similar law than with the other cases, tending to increase along with the tasks flexibility. For low flexibility factors, the two weighted sum methods give much more violations than the fuzzy-IT one, or the modified GreenSlot algorithm.

As we take into account the grid energy price with this electrical attractiveness function, we note that the grid energy consumption (Fig. 10(b)) and cost of this energy (Fig. 10(c)) are much more similar than for the variant A.

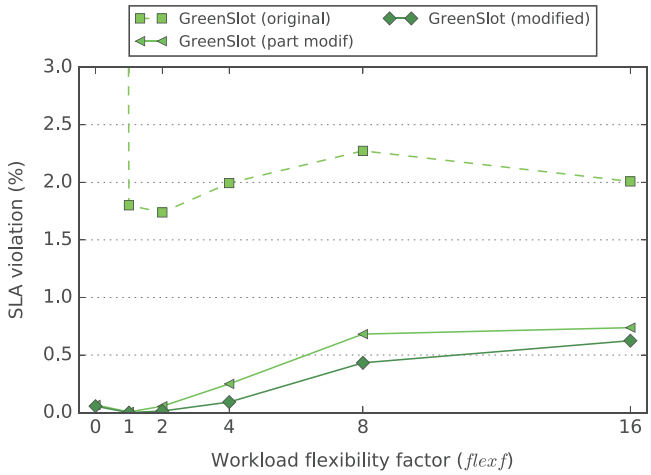
5.3. Comparison between approaches

Fig. 11 summarizes the results for most of the presented approaches and heuristics. The results of ABBSH using the weighted sum of hyperbolic sinus as multi-objective methods are not shown as they differ only slightly from using the simple weighted sum. Three flexibility factors are considered, 2, 8 and 16, which seem representative of workloads with respectively a low, medium and high level of flexibility. In this figure, the results obtained by using a first-fit scheduler are also represented, illustrating a traditional scheduling algorithm with no renewable energy consideration.

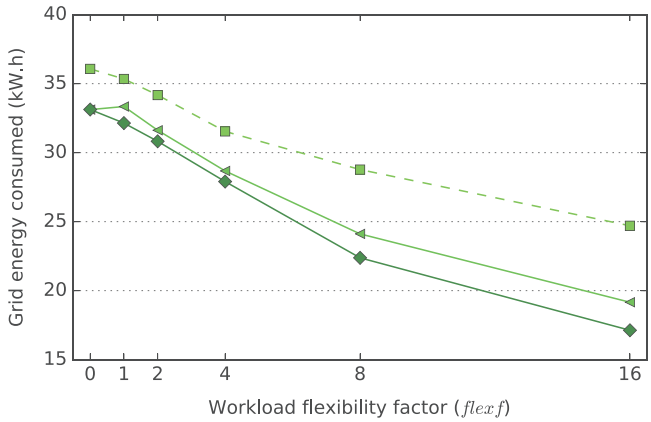
For workloads with low flexibility, our approach using the fuzzy-based method outperforms the GreenSlot one, even with all our modifications. Whereas, for *flexf* $= 2$, the GreenSlot *modified* version reduces the use of grid energy by 7.6% compared to a first-fit scheduling, our approach, with the fuzzy-IT method and the variant B of electrical attractiveness function, reaches 10.5%. The cost saving is even more important, with only 9.6% for GreenSlot compared to 12.4% for fuzzy-IT.

The same heuristic (fuzzy-IT with variant B) gives results similar to the *modified* version of GreenSlot for the workloads with high flexibility. The GreenSlot scheduler reaches 48.2% grid energy saving and 51% cost saving. Our ABBSH algorithm performs slightly better on this scenario, with a grid energy saving of 49.4% and a reduction of 51.2% of the cost, compared to the traditional scheduler.

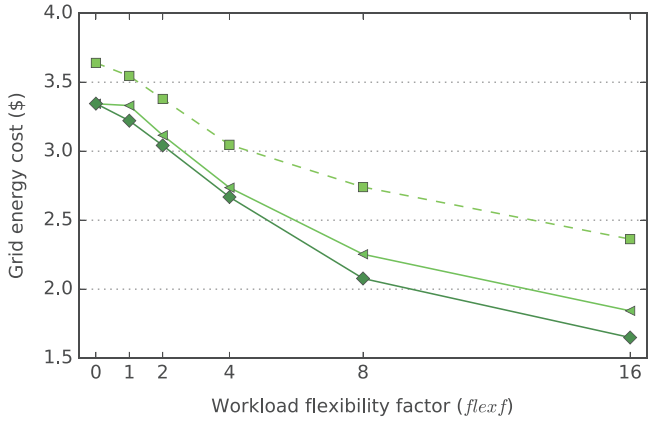
Our approach is slightly outperformed for workloads with medi um flexibility (*flexf* $= 8$). In this case, our best heuristic for



(a) SLA violation rate. The data point for GreenSlot original with a flexibility factor of 0 is not visible with this scale (its value is 98%).

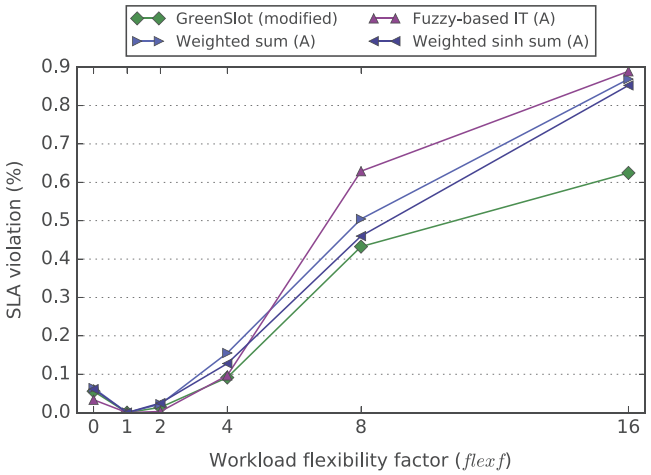


(b) Energy consumption from the grid.

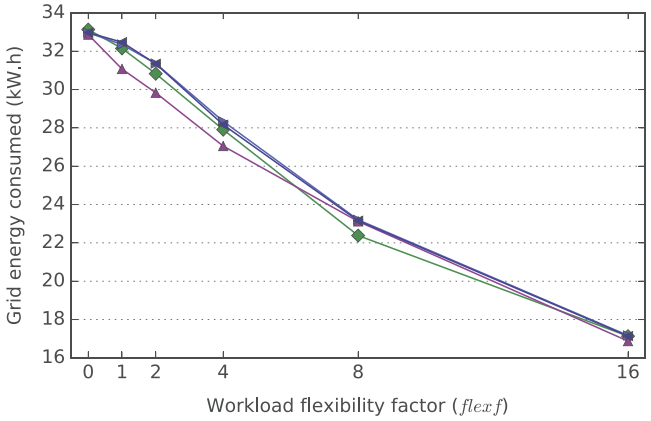


(c) Cost of the grid energy.

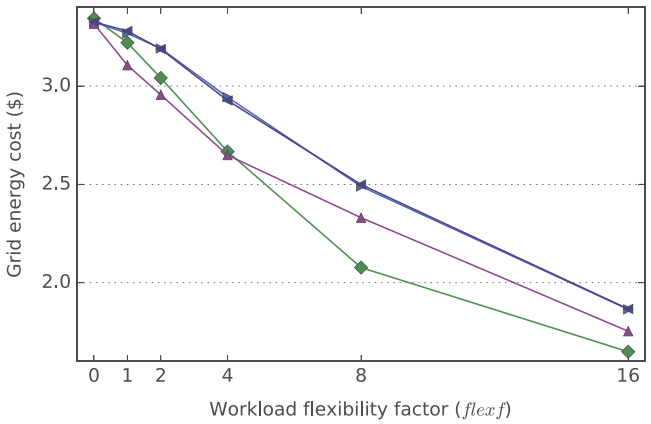
Fig. 8. Impact of flexibility factor over several metrics, for different versions of the GreenSlot scheduler implemented.



(a) SLA violation rate (percentage of the total task number).



(b) Energy consumption from the grid.



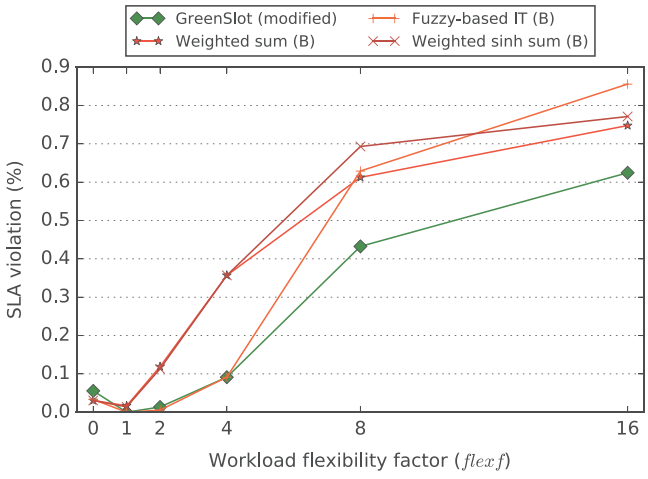
(c) Cost of the grid energy.

Fig. 9. Impact of flexibility factor over several metrics, for ABBSH with different multi-objective functions, using variant A of electrical attractiveness function (without consideration of grid energy price).

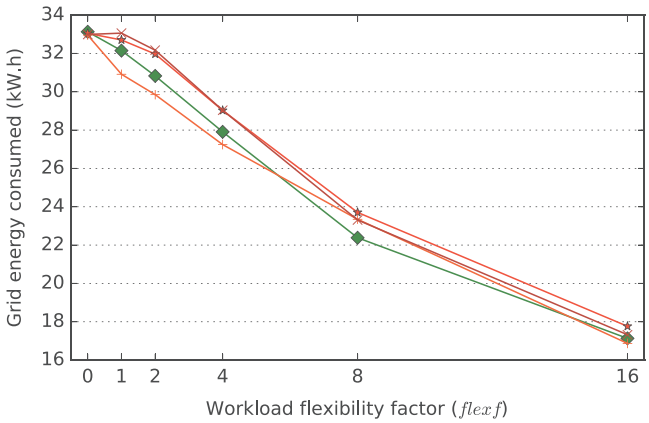
reducing non-renewable energy use is the fuzzy-IT with variant A, reaching 31% saving, and the best one for reducing cost, with 35.4% saving, is the simple weighted sum with variant B. The GreenSlot *modified* version reduces the grid consumption by 33.2% and the cost by 38.6%.

In all the scenarios, the *original* version of GreenSlot gives the worst results, showing again that it is not adapted to the kind of workload we used. Notably, when the flexibility factor is low, it actually increases both non-renewable energy use (+2.5% for $flexf = 2$) and total cost (+0.4%) compared to a first-fit scheduling. In addition, it causes important SLA violation rate compared to the other heuristics, as shown in Fig. 11(a).

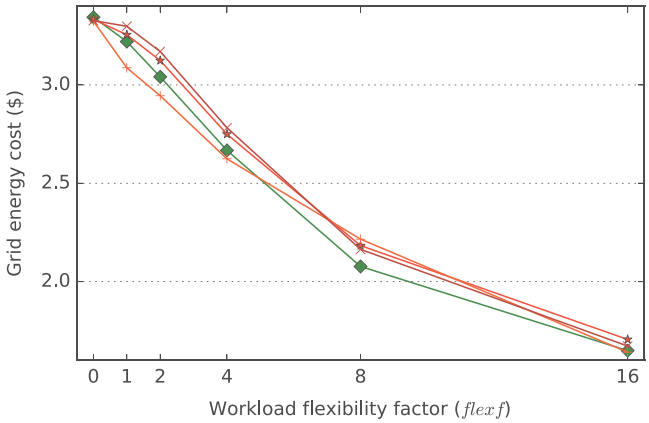
The other heuristics give an average SLA violation rate lower than 1% on each of the three scenarios. However, at the exception of case of workloads with low flexibility, the violation rate of our



(a) SLA violation rate (percentage of the total task number).

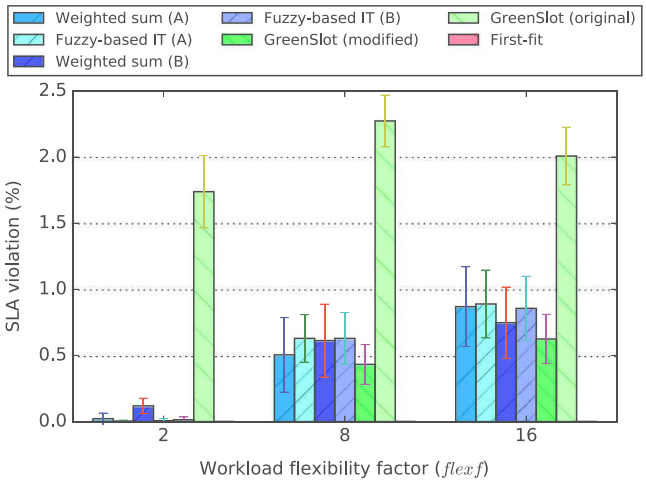


(b) Energy consumption from the grid.

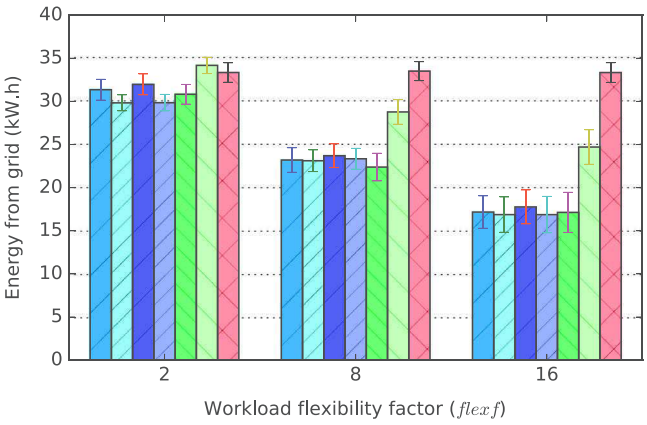


(c) Cost of the grid energy.

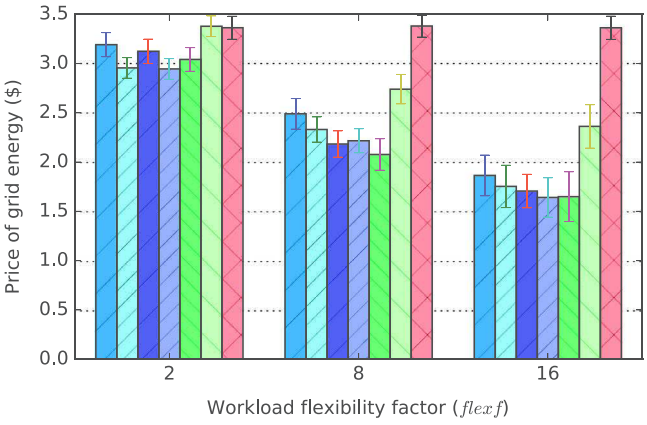
Fig. 10. Impact of flexibility factor over several metrics, for ABBSH with different multi-objective functions, using variant B of electrical attractiveness function (consideration of the variation of the grid energy price).



(a) SLA violation rate (percentage of the total task number). First-fit is always at 0, and therefore not visible.



(b) Energy consumption from the grid.



(c) Cost of the grid energy.

Fig. 11. Comparison of the results between the approaches and heuristics, for workloads with different flexibility values. The standard deviation is represented, as each experiment is repeated with 10 workloads generated using different random seeds.

heuristics is slightly higher than the one of the GreenSlot version with our modifications.

5.4. Comparison to the lower bound

Based on the method presented in Section 4.5, we computed a lower bound of the grid energy consumption for every workload used in the previous experiments. The results for different values of the flexibility factor are given in Fig. 12(a), along with one of the ABBSH heuristic and the two variants of GreenSlot. To interpret those results, it is important to emphasize that the model used for this lower bound does not take into account some of the energy

consumed normally. Notably, the energy needed to power on and off the machines is not accounted.

Therefore, the total energy consumed (both from grid and renewable sources) is significantly less than with the tested heuristics. For all of them, this amount of total energy varies little depending on the flexibility. The average, for each of the variants of our approach as well as for the modified GreenSlot algorithm, are between 55.7 kWh and 56.6 kWh. This value is as high as 60.5 kWh for the original GreenSlot variant. By contrast, the solutions from the lower bound method give an energy consumption of 50.4 kWh.

The shape of the grid energy consumed curves, from Fig. 12(a), are similar for the different heuristics and for the lower bound. The absolute difference between the lower bound and the best result of all the heuristics increases with the flexibility factor for small values (from 3.8 kWh for $flexf = 0$ to 5.4 kWh for $flexf = 4$). For higher flexibility factors, the difference is almost constant (5.6 kWh and 5.4 kWh respectively for $flexf = 8$ and $flexf = 16$).

Fig. 12(b) shows the percentage of the energy consumed provided by the renewable sources. The method used to compute the lower bound of the grid consumption does not give an upper bound to the percentage of renewable energy used. However, those values help to take into account the differences in total energy consumption.

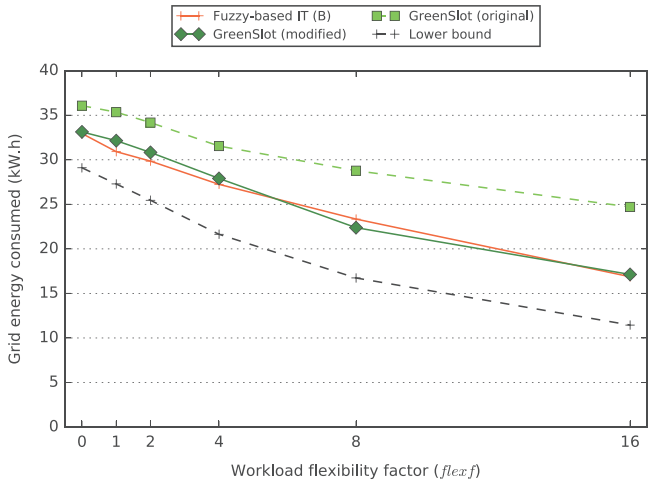
The lower bound shown in Figs. 12(a) and 12(b), demonstrates that the heuristic we propose provides gain of the same order of magnitude of what could possibly be reachable under our hypothesis.

6. Discussion

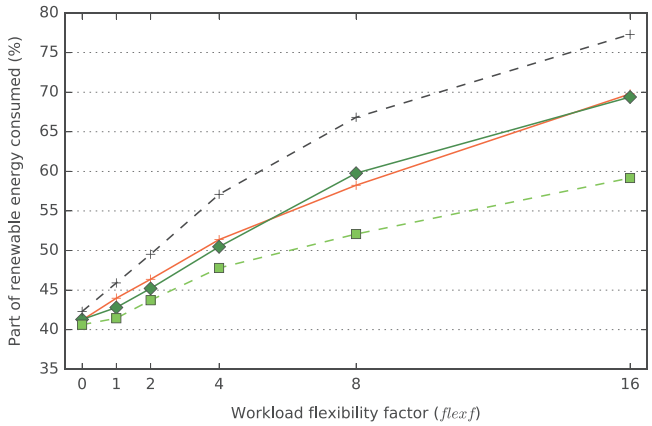
One of the interesting findings of our experiments concerns the impact, on the performances of the tested heuristics, of workload flexibility (e.g. the average time available for scheduling each task and meeting their due dates). While the increase of flexibility allows, as expected, to improve the results in terms of non-renewable energy and cost saving, it also leads to an increase of SLA violations. It may seem counter-intuitive, but is easily explained by the increased usage of computing resources when the energy is available. As some tasks, with important flexibility, are delayed to be executed at those moments, a task with low flexibility submitted when all machines are already reserved is likely to be scheduled after its due date. It may be possible to overcome this issue by using a heuristic which takes into account a prediction of tasks submitted in the future, in order to keep some computing resources available for them.

The relationship between the workload flexibility and the reduction of the cost and of the grid energy use seems quite complex. Beyond a certain flexibility ($flexf > 8$ in our experiments), increasing it still allows to give better results, but proportionally less important to what is achievable by increasing it as much below this threshold. This finding is reinforced by the results of our lower bound of grid consumption. The saving of grid energy as a function of the flexibility factor behave almost linearly for $flexf \in [0, 4]$, as shown in Fig. 12. However, we believe that it is particularly related to the profile of the renewable source we used. Indeed, our simulated solar panels give a very regular power, as if every day was perfectly sunny. If, instead, we had simulated a more realistic weather, a cloudy day would involve to move as much tasks as possible to the next day, and therefore would probably take advantage of tasks with very high flexibility.

Contrary to other studies, like [12,13], our work shows how the potential energy saving of a renewable-aware scheduler is constrained by the SLA negotiated with the users. Also, we believe it is important to define a metric which gives the amount of freedom for the scheduling of a workload. The definition of this metric would depend on the kind of workload considered. In the case of a batch



(a) Energy consumption from the grid.



(b) Portion of renewable energy in the total consumption.

Fig. 12. Comparison of the results between different approaches and the lower bound of grid consumption.

tasks with due dates, a more generic equivalent of our *flexibility factor* may be used (for example a measurement of the average or median of the flexibility of the tasks). We see two major reasons for defining such metrics. Firstly it would make possible to compare more easily the performances of different approaches, and to evaluate a given approach with several well identified hypothesis of SLA level. Secondly it would allow to measure it in real data centers, either to predict the performances of a given approach for this specific data center, or to use this measurement as a reference for future researches.

The aforementioned relationship between SLA and potential energy saving could also be used to study possible *green pricing* for data center operators. The operators can promote lower SLA level to the users, by the way of incentive schemes, in order to save energy and money. Such incentives may concern the price of the jobs, but not necessarily limited to that. Indeed, other parameters also play a role in the energy saving behavior of individuals [47].

The results of the several variants of our ABBSH algorithm show that all of them reduce both grid energy usage and energy cost compared to a traditional scheduler. Taking the grid energy price into account (with the electrical attractiveness variant B) leads to a small increase of non-renewable energy use, but allows an important reduction of the grid energy cost, as illustrated by Figs. 11(b) and 11(c). The *fuzzy-based* multi-objective method provides, in overall, the most interesting trade-off between respect of SLA and energy or cost saving.

By introducing some modifications into the GreenSlot scheduler [12], we adapted it to the kind of workload we used for validating our approach, as evidenced by the results from Fig. 8. In the original article, the authors used a very grid-oriented scientific workload. In contrast, the workload we used, based on the traces of a Google cloud cluster, is more representative of a typical batch workload executed in cloud infrastructures, like map-reduce jobs or periodic indexing tasks. With our modifications, the performances of GreenSlot are quite similar with those of ABBSH. Whereas GreenSlot is a bit more efficient than our heuristics with a medium flexibility, our approach gives slightly better results for workloads with low flexibility ($flexf \leq 4$).

A notable fact is the proximity of the results of both approaches. For every flexibility factor value, and for both grid energy consumption and cost, the results of our best heuristic and those of the modified version of GreenSlot are included into a 5% interval. We believe that such close results for two different heuristic-based approaches suggest they are near to a local optimum of this kind of online and greedy heuristics. The results of the proposed lower bound gives only a partial answer. Looking to the absolute difference of the grid consumption with the tested heuristics and to the percentage of renewable energy used, it seems possible to slightly improve our approach. Unfortunately, we are currently unable to tell exactly how much improvement can be done, as this lower bound does not take some of the characteristics of the problem into account (such as the cost of powering on and off the machines). Trying to solve the formal optimization problem, at least for small instances, would allow to identify exactly the improvement margin we can expect.

The ABBSH algorithm we propose has only a partial knowledge of the electrical infrastructure, provided by the results of the attractiveness function. By reaching results close to GreenSlot, an approach which requires a total knowledge of the electrical model, our work shows that it is possible to use efficiently the renewable energy in a data center with separated (electrical and IT) management systems. However, with the presented approach, both systems simply *cooperate* with the other, and the scheduling algorithm is not designed to minimize the amount of communication needed to find a solution.

7. Conclusion and future work

In this paper, we presented ABBSH, a new algorithm with several variants for scheduling batch tasks with awareness of renewable energy and electricity cost. Our scheduler, contrary to other approaches, never deals directly with the model of the electrical infrastructure, but instead uses partial and abstracted information. By using those hints to guide the scheduler, our approach gives results close to, or slightly better, to another scheduler of the literature which has a full knowledge of the electrical model.

Moreover, our results give insights on the relationship between the magnitude of the SLA negotiated for the tasks of the workload, and the performances of the different heuristics studied. With SLA allowing more freedom on the time a submitted task should be executed, a renewable-aware scheduler can take more efficiently advantage of the intermittent nature of most renewable energy sources. As an indirect consequence, this increases the part of the computing resources used when the energy is the more available, leading to an inability to satisfy stricter SLAs of tasks submitted at these times.

The performances of the different heuristics are also compared to a lower bound of the grid energy consumption. The results show that our approach performs already well and gives an idea of maximum margin of improvement. In addition, such lower bound formulation with lower complexity than the real optimization problem can be used to continue to investigate on the relationship between SLA and possible cost and energy savings.

Our future works will focus on three main areas. Firstly, as stated previously, we would like to compare the performances of our heuristics with the optimal schedule. For this purpose, we plan to solve a MILP formulation of the optimization problem. This may also help to figure out situations where our scheduling algorithm is far from the optimal, giving some clues to improve it. Secondly, we are currently working in improving and integrating our modifications into DCworms, the data center simulator we used for our experiments. By providing a data center simulator able to simulate a complex electrical infrastructure, we hope to make it easier for other researchers to study scheduling policies with on-site renewable energy sources. Finally, another direction of our work is to go beyond the *cooperation* and to explore further the decentralized aspect of our scheduling algorithm. Indeed, we would like to develop an approach based on a collaborative *negotiation* between electrical and IT management systems, and to study the relationship between the amount of information exchanged and the quality of the resulting solution.

As the proposed work is complementary to a large number of approaches using broader points of view, there are also a large number of opportunities to add external elements to the model such as a cooling infrastructure or other energy storage systems such as full cells, along with specific optimizations for these elements.

Acknowledgments

The work presented in this paper has been funded by the ANR in the context of the project DATAZERO, ANR-15-CE25-0012. Experiments presented in this paper were carried out on Grid5000 testbed, supported by a scientific interest group hosted by Inria, including CNRS, RENATER, several Universities and organizations (http://www.grid5000.fr).

References

[1] W. Van Heddeghem, S. Lambert, B. Lannoo, D. Colle, M. Pickavet, P. Demeester, Trends in worldwide ICT electricity consumption from 2007 to 2012, Comput. Commun. (ISSN: 0140-3664) 50 (2014) 64–76. http://dx.doi.org/10.1016/j.comcom.2014.02.008.

[2] A.S.G. Andrae, T. Edler, On global electricity usage of communication technology: Trends to 2030, Challenges 6 (1) (2015) 117–157. http://dx.doi.org/10.3390/challe6010117.

[3] J. Miller, L. Bird, J. Heeter, B. Gorham, Renewable Electricity Use by the U.S. Information and Communication Technology (ICT) industry, Tech. Rep. NREL/TP-6A20-64011, National Renewable Energy Lab. (NREL), Golden, CO, United States, 2015. http://dx.doi.org/10.2172/1215195.

[4] L.-D. Radu, Determinants of green ICT adoption in organizations: A theoretical perspective, Sustainability 8 (8) (2016) 731. http://dx.doi.org/10.3390/su8080731.

[5] V. Krakowski, E. Assoumou, V. Mazauric, N. Maïzi, Reprint of feasible path toward 40–100% renewable energy shares for power supply in France by 2050: A prospective analysis, Appl. Energy (ISSN: 0306-2619) 184 (2016) 1529–1550. http://dx.doi.org/10.1016/j.apenergy.2016.11.003.

[6] B. Elliston, I. MacGill, M. Diesendorf, Least cost 100% renewable electricity scenarios in the Australian National Electricity market, Energy Policy (ISSN: 0301-4215) 59 (2013) 270–282. http://dx.doi.org/10.1016/j.enpol.2013.03.038.

[7] Z. Liu, M. Lin, A. Wierman, S.H. Low, L.L. Andrew, Geographical load balancing with renewables, SIGMETRICS Perform. Eval. Rev. (ISSN: 0163-5999) 39 (3) (2011) 62–66. http://dx.doi.org/10.1145/2160803.2160862.

[8] D. Hatzopoulos, I. Koutsopoulos, G. Koutitas, W.V. Heddeghem, Dynamic virtual machine allocation in cloud server facility systems with renewable energy sources, in: 2013 IEEE International Conference on Communications, ICC, 2013, pp. 4217–4221. http://dx.doi.org/10.1109/ICC.2013.6655225.

[9] A. Khosravi, A. Nadjaran Toosi, R. Buyya, Online virtual machine migration for renewable energy usage maximization in geographically distributed cloud data centers, Concurr. Comput.: Pract. Exper. (ISSN: 1532-0634) (2017). http://dx.doi.org/10.1002/cpe.4125.

[10] Z. Liu, Y. Chen, C. Bash, A. Wierman, D. Gmach, Z. Wang, M. Marwah, C. Hyser, Renewable and cooling aware workload management for sustainable data centers, in: Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '12, ACM, New York, NY, USA, ISBN: 978-1-4503-1097-0, 2012, pp. 175–186. http://dx.doi.org/10.1145/2254756.2254779.

[11] B. Aksanli, J. Venkatesh, L. Zhang, T. Rosing, Utilizing green energy prediction to schedule mixed batch and service jobs in data centers, SIGOPS Oper. Syst. Rev. (ISSN: 0163-5980) 45 (3) (2012) 53–57. http://dx.doi.org/10.1145/2094091.2094105.
[12] Í. Goiri, K. Le, M.E. Haque, R. Beauchea, T.D. Nguyen, J. Guitart, J. Torres, R. Bianchini, GreenSlot : Scheduling energy consumption in green datacenters, in: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, SC '11, ACM, New York, NY, USA, ISBN: 978-1-4503-0771-0, 2011, pp. 20:1–20:11. http://dx.doi.org/10.1145/2063384.2063411.
[13] Í. Goiri, W. Katsak, K. Le, T.D. Nguyen, R. Bianchini, Parasol and GreenSwitch: Managing datacenters powered by renewable energy, in: Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '13, ACM, New York, NY, USA, ISBN: 978-1-4503-1870-9, 2013, pp. 51–64. http://dx.doi.org/10.1145/2451116.2451123.
[14] A. Krioukov, S. Alspaugh, P. Mohan, S. Dawson-Haggerty, D.E. Culler, R.H. Katz, Design and Evaluation of an Energy Agile Computing Cluster, Tech. Rep. UCB/EECS-2012-13, EECS Department, University of California, Berkeley, 2012.
[15] Y. Li, A.C. Orgerie, J.M. Menaud, Balancing the use of batteries and opportunistic scheduling policies for maximizing renewable energy consumption in a cloud data center, in: 2017 25th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP, 2017, pp. 408–415. http://dx.doi.org/10.1109/PDP.2017.24.
[16] B. Jennings, R. Stadler, Resource management in clouds: Survey and research challenges, J. Netw. Syst. Manage. 23 (3) (2015) 567–619. http://dx.doi.org/10.1007/s10922-014-9307-7. (ISSN: 1064-7570, 1573-7705).
[17] L. Furtado, M. Dutra, D. Macedo, Value creation in big data scenarios: A literature survey, J. Ind. Integr. Manag. 02 (01) (2017) 1750002. http://dx.doi.org/10.1142/S2424862217500026.
[18] T. Guérout, S. Medjiah, G. Da Costa, T. Monteil, Quality of service modeling for green scheduling in clouds, Sustain. Comput. Inform. Syst. (ISSN: 2210-5379) 4 (4) (2014) 225–240. http://dx.doi.org/10.1016/j.suscom.2014.08.006.
[19] T. Guérout, Y. Gaoua, C. Artigues, G. Da Costa, P. Lopez, T. Monteil, Mixed integer linear programming for quality of service optimization in clouds, Future Gener. Comput. Syst. (ISSN: 0167-739X) 71 (2017) 1–17. http://dx.doi.org/10.1016/j.future.2016.12.034.
[20] J. Shuja, K. Bilal, S.A. Madani, M. Othman, R. Ranjan, P. Balaji, S.U. Khan, Survey of techniques and architectures for designing energy-efficient data centers, IEEE Syst. J. (ISSN: 1932-8184) 10 (2) (2016) 507–519. http://dx.doi.org/10.1109/JSYST.2014.2315823.
[21] L. Wang, G. von Laszewski, J. Dayal, F. Wang, Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS, in: 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGrid, 2010, pp. 368–377. http://dx.doi.org/10.1109/CCGRID.2010.19.
[22] G. von Laszewski, L. Wang, A.J. Younge, X. He, Power-aware scheduling of virtual machines in DVFS-enabled clusters, in: 2009 IEEE International Conference on Cluster Computing and Workshops, 2009, pp. 1–10. http://dx.doi.org/10.1109/CLUSTR.2009.5289182.
[23] Z. Zhou, J. Abawajy, M. Chowdhury, Z. Hu, K. Li, H. Cheng, A.A. Alelaiwi, F. Li, Minimizing SLA violation and power consumption in cloud data centers using adaptive energy-aware algorithms, Future Gener. Comput. Syst. (ISSN: 0167-739X) (2017). http://dx.doi.org/10.1016/j.future.2017.07.048.
[24] H. Sun, P. Stolf, J.-M. Pierson, Spatio-temporal thermal-aware scheduling for homogeneous high-performance computing datacenters, Future Gener. Comput. Syst. (ISSN: 0167-739X) 71 (Supplement C) (2017) 157–170. http://dx.doi.org/10.1016/j.future.2017.02.005.
[25] X. Li, X. Jiang, P. Garraghan, Z. Wu, Holistic energy and failure aware workload scheduling in cloud datacenters, Future Gener. Comput. Syst. (ISSN: 0167-739X) 78 (Part 3) (2018) 887–900. http://dx.doi.org/10.1016/j.future.2017.07.044.
[26] Z. Li, J. Ge, C. Li, H. Yang, H. Hu, B. Luo, V. Chang, Energy cost minimization with job security guarantee in internet data center, Future Gener. Comput. Syst. (ISSN: 0167-739X) 73 (Supplement C) (2017) 63–78. http://dx.doi.org/10.1016/j.future.2016.12.017.
[27] F. Kong, X. Liu, A survey on green-energy-aware power management for datacenters, ACM Comput. Surv. (ISSN: 03600300) 47 (2) (2014) 1–38. http://dx.doi.org/10.1145/2642708.
[28] C. Gu, L. Fan, W. Wu, H. Huang, X. Jia, Greening cloud data centers in an economical way by energy trading with power grid, Future Gener. Comput. Syst. (ISSN: 0167-739X) 78 (Part 1) (2018) 89–101. http://dx.doi.org/10.1016/j.future.2016.12.029.
[29] Y. Li, A.C. Orgerie, J.M. Menaud, Opportunistic scheduling in clouds partially powered by green energy, in: 2015 IEEE International Conference on Data Science and Data Intensive Systems, 2015, pp. 448–455. http://dx.doi.org/10.1109/DSDIS.2015.80.
[30] C. Chen, S. Duan, T. Cai, B. Liu, Online 24-h solar power forecasting based on weather type classification using artificial neural network, Sol. Energy (ISSN: 0038-092X) 85 (11) (2011) 2856–2870. http://dx.doi.org/10.1016/j.solener.2011.08.027.
[31] E. İzgi, A. Öztopal, B. Yerli, M.K. Kaymak, A.D. Şahin, Short–mid-term solar power prediction by using artificial neural networks, Sol. Energy (ISSN: 0038-092X) 86 (2) (2012) 725–733. http://dx.doi.org/10.1016/j.solener.2011.11.013.
[32] G. Sideratos, N.D. Hatziargyriou, An advanced statistical method for wind power forecasting, IEEE Trans. Power Syst. (ISSN: 0885-8950) 22 (1) (2007) 258–265. http://dx.doi.org/10.1109/TPWRS.2006.889078.
[33] S. Li, D.C. Wunsch, E.A. O'Hair, M.G. Giesselmann, Using neural networks to estimate wind turbine power generation, IEEE Trans. Energy Convers. (ISSN: 0885-8969) 16 (3) (2001) 276–282. http://dx.doi.org/10.1109/60.937208.
[34] V. Villebonnet, G.D. Costa, L. Lefevre, J.M. Pierson, P. Stolf, Energy aware dynamic provisioning for heterogeneous data centers, in: 2016 28th International Symposium on Computer Architecture and High Performance Computing, SBAC-PAD, 2016, pp. 206–213. http://dx.doi.org/10.1109/SBAC-PAD.2016.34.
[35] H. Sun, P. Stolf, J.-M. Pierson, G. Da Costa, Energy-efficient and thermal-aware resource management for heterogeneous datacenters, Sustain. Comput. Inform. Syst. (ISSN: 2210-5379) 4 (4) (2014) 292–306. http://dx.doi.org/10.1016/j.suscom.2014.08.005.
[36] K. Kurowski, A. Oleksiak, W. Piątek, T. Piontek, A. Przybyszewski, J. Węglarz, DCworms — a tool for simulation of energy efficiency in distributed computing infrastructures, Simul. Model. Pract. Theory (ISSN: 1569-190X) 39 (2013) 135–151. http://dx.doi.org/10.1016/j.simpat.2013.08.007.
[37] L.F. Cupertino, G.D. Costa, A. Oleksiak, W. Piatek, J.-M. Pierson, J. Salom, L. Sisó, P. Stolf, H. Sun, T. Zilio, Energy-efficient, thermal-aware modeling and simulation of datacenters: The CoolEmAll approach and evaluation results, ResearchGate (ISSN: 1570-8705) 25 (B) (2015) 535–553. http://dx.doi.org/10.1016/j.adhoc.2014.11.002.
[38] G. Da Costa, L. Grange, I. De Courchelle, Modeling and generating large-scale Google-like workload, in: 2016 Seventh International Green and Sustainable Computing Conference (IGSC), Hangzhou, 2016, pp. 1–7. http://dx.doi.org/10.1109/IGCC.2016.7892623.
[39] E. Jones, T. Oliphant, P. Peterson, SciPy: Open source scientific tools for Python, 73, 2001, p. 86. URL http://www.scipy.org.
[40] V. Villebonnet, G.D. Costa, L. Lefevre, J.M. Pierson, P. Stolf, Dynamically building energy proportional data centers with Heterogeneous Computing Resources, in: 2016 IEEE International Conference on Cluster Computing, CLUSTER, 2016, pp. 217–220. http://dx.doi.org/10.1109/CLUSTER.2016.34.
[41] F. Strunk, An Analysis of Linux Boot Times (Ph.D. thesis), Chemnitz University of Technology, 2008.
[42] A.C. Orgerie, L. Lefèvre, J.P. Gelas, Demystifying energy consumption in grids and clouds, in: 2010 International Green Computing Conference, 2010, pp. 335–342. http://dx.doi.org/10.1109/GREENCOMP.2010.5598295.
[43] Y. Chu, P. Meisen, Review and Comparison of Different Solar Energy Technologies, Global Energy Network Institute (GENI), San Diego, CA, 2011.
[44] D.D. Chiras, Power from the Sun: A Practical Guide to Solar Electricity, New Society Publishers, ISBN: 978-1-55092-433-6, 2013.
[45] P. Ranganathan, P. Leech, D. Irwin, J. Chase, Ensemble-level power management for Dense Blade servers, in: Proceedings of the 33rd Annual International Symposium on Computer Architecture, ISCA '06, IEEE Computer Society, Washington, DC, USA, ISBN: 978-0-7695-2608-9, 2006, pp. 66–77. http://dx.doi.org/10.1109/ISCA.2006.20.
[46] J.D. Ullman, NP-complete scheduling problems, J. Comput. System Sci. (ISSN: 0022-0000) 10 (3) (1975) 384–393. http://dx.doi.org/10.1016/S0022-0000(75)80008-0.
[47] Y. Fan, Research on factors influencing an individual's behavior of energy management: A field study in China, J. Manag. Anal. (ISSN: 2327-0012) 4 (3) (2017) 203–239. http://dx.doi.org/10.1080/23270012.2017.1310000.

Léo Grange is a Ph.D. student in Computer Science at the Institut de Recherche en Informatique de Toulouse (IRIT), at the University of Toulouse. He has completed his Master's degree in Computer Science, specialized in Distributed Systems and Safety Critical Software, at the University of Toulouse.

For his Ph.D. studies, he is working on energy-efficient scheduling for cloud computing, and more specifically for data centers powered with on-site renewable energy sources. He is actively involved in the ANR Datazero project, where he is studying the possibilities of collaboration between the management of multiple green energy sources and the management of IT resources and services.

Georges Da Costa is Assistant Professor in Computer Science at the University of Toulouse. He received its Ph.D. from LIG (Grenoble, France) in 2005. He is a member of the IRIT Laboratory. His main interests are related to large-scale distributed systems, algorithmics, performance evaluation and energy-aware systems.

He serves on several PCs in the Energy aware systems, Grid and Peer to Peer fields. He is chair of the COST1305 working group 2 on 'Programming models and runtimes'. His research currently focus on energy aware distributed systems. His research highlights are grid cluster & cloud computing, peer to peer, large scale energy aware distributed systems, performance evaluation, ambient systems.

Patricia Stolf is an Associate Professor in Computer Science at the University of Toulouse. She received her Ph.D. from INSA (Toulouse, France), in 2004. She is a member of the IRIT Laboratory. Her main interests are related to large scale distributed systems like grid or clouds, distributed algorithms and autonomic computing. Her research currently focuses on resources management, load-balancing, energy aware and thermal aware placement.

She has been involved in different research projects: in the ACTION COST IC0804 Energy Efficiency in Large Scale Distributed Systems, in the European CoolEmAll project and in the national ANR SOP project.

She is currently working on the ANR Datazero project studying how to manage the electricity and IT services in a datacenter operated with several green energy sources.